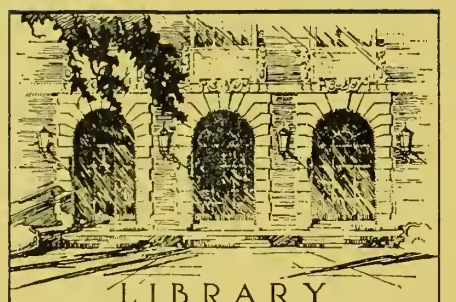


Report No. 230

COO-1018-1115

THE THEORY AND IMPLEMENTATION OF SRT DIVISION

by

Daniel E. Atkins III

June 1, 1967

DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

Report No. 230

THE THEORY AND IMPLEMENTATION OF SRT DIVISION

by

Daniel E. Atkins III

June 1, 1967

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

*This work was submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering, June 1967, and was supported in part by the AEC under Contract No. USAEC AT(11-1)1018.

ACKNOWLEDGEMENT

I wish to thank Professor S. R. Ray for his most helpful advice and assistance in the preparation of this report. I also thank Professor J. E. Robertson for the enlightening discussions concerning the material in Chapter 2.

I further acknowledge and thank Mr. Richard Borovec for his discussions concerning the cost determinations (Section 2.6), Mrs. L. A. Prendergast and Mr. Ronald C. Morrison for the drawings, and Mrs. Anita Worthington for the typing of the final draft.

TABLE OF CONTENTS

1. INTRODUCTION

Perhaps the major complication associated with digital division is best illustrated by your performing the following long-division problem and noting carefully the steps you follow.

$$396_{\Delta} \overline{\big| 1057_{\Delta_1} 6_{\Delta_2} 2_{\Delta_3} 1}$$

Δ = decimal point marker

Your operations in selecting the first quotient digit are summarized in the flow chart, Figure 1. The salient point is that division is a trial and error process requiring an initial "guess" of a quotient digit followed by a subtraction, or at least a comparison, to determine whether the guess is correct. If it is not, the initial choice is modified and the process repeated. It is the trial and error nature of division, whether performed by man or machine, which complicates its execution. In building a computer arithmetic unit, division is the most difficult basic operation to implement efficiently.

But despite the complexity, the literature is replete with themes and variations for implementing digital division. Flores, [1]* for example, states four methods for increasing speed of division and then proceeds to describe no less than twenty-four schemes which incorporate some or all of these speed-up techniques. MacSorley [2] describes four division techniques demanding various divisor multiples to accelerate execution.

*Numbers in brackets refer to the corresponding entry under References.

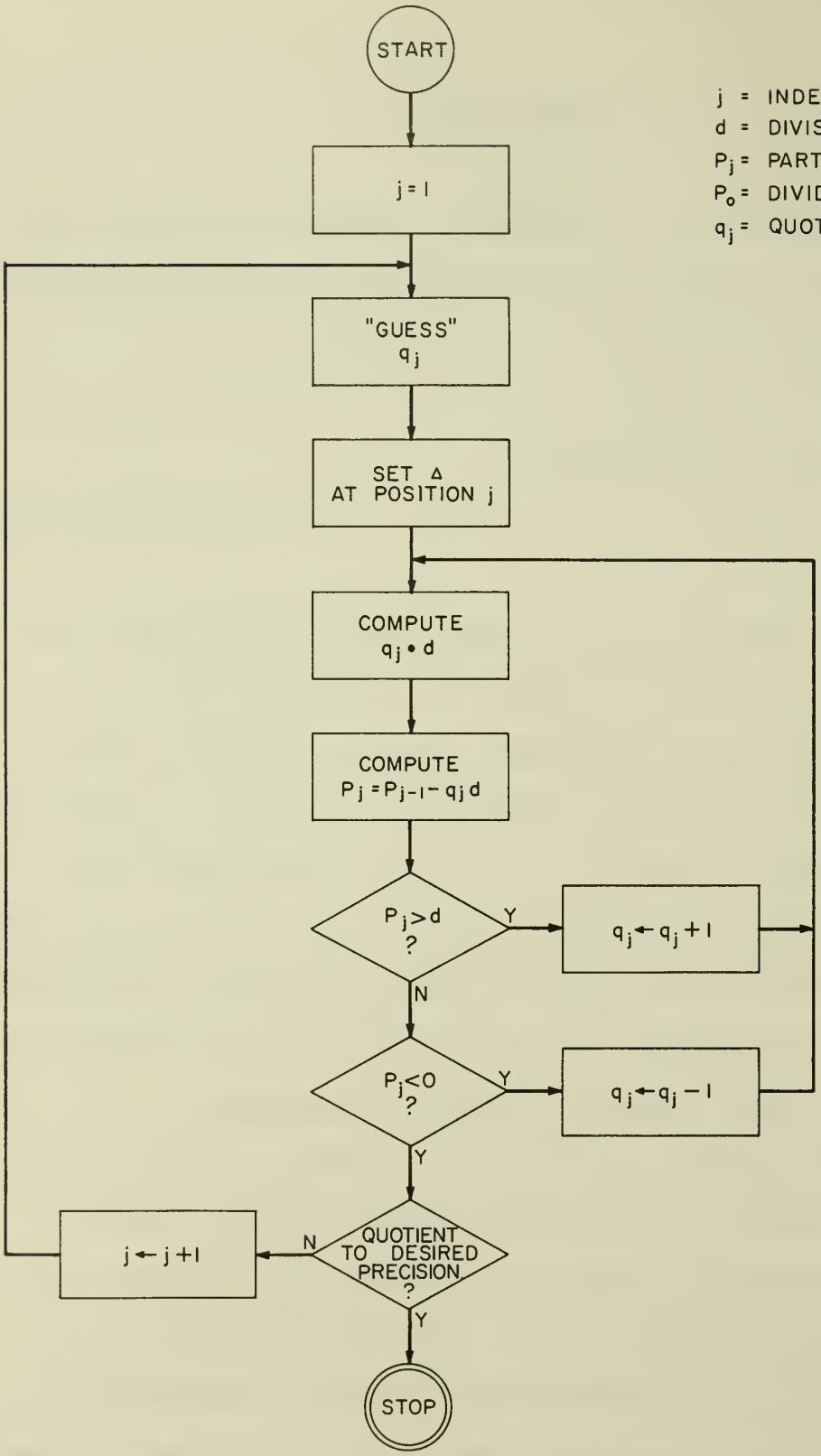


FIGURE I. FLOWCHART OF MANUAL EXECUTION OF DIVISION

There is far less in the literature, however, describing theory and analytic tools to be used in designing a division scheme. Most of the articles describe schemes which are products more of art than of science. This report is an attempt to contribute to the science of computer arithmetic implementation.

This report describes a class of division techniques especially suited for implementation in an electronic digital computer. For historic reasons, this class will be referred to as SRT division. The name is derived from the fact that the binary case of this type of division was discovered independently, at about the same time, by Dura Sweeney of IBM, J. E. Robertson of the University of Illinois, and T. D. Tocher of Imperial College, London [3]. The paper, however, incorporates more recent work, due exclusively to Professor Robertson, which extends the binary SRT division to a radix higher than two. Much of Chapter 2 is based upon his report [5] and upon numerous personal communications.

After a description of the theory and properties of SRT division, the report turns to the problem of actually implementing the scheme and presents an example of one possible realization.

2. THE THEORY OF SRT DIVISION

2.0 Introduction

This chapter introduces a recursive relationship for describing division and from it develops the nature of SRT division. The discussion is augmented with two graphical representations; one to determine the range restrictions associated with SRT, and the other to aid in computing the "cost" of quotient digit selection.

Most of the following analysis will be developed for a general radix, r. At first this generality may appear superfluous, for after all, isn't a digital computer a binary machine, and doesn't binary imply radix two? It is true that the basic storage elements of a digital computer are two state devices and that numbers are represented internally by strings of "1's" and "0's". Computer arithmetic, however, is often facilitated by considering groups of bits rather than each bit individually. Such grouping may be interpreted as use of digits of higher radix than two. For example, a pair of bits becomes one, radix four digit; a trio of bits, a radix eight (octal) digit.

In the literature of arithmetic unit design, one finds references to such techniques as inspection of bits "two at a time," or perhaps "generation of several quotient bits simultaneously". In this report such techniques would be described in terms of higher radix arithmetic.

2.1 The Recursive Relationship

Digital division as implemented in an electronic computer consists of preliminary operations, i.e., normalization, a recursive process, and a terminal operation, i.e., changing the form of the remainder. Although preliminary and terminal operations vary from machine to machine, they generally consume much less of the execution time than the recursive operations. For restoring, non-restoring, and the SRT division scheme to be described in this report, this recursive relationship is defined by

$$p_{j+1} = rp_j - q_{j+1} d \qquad (2.1.1)$$

where the symbols are defined as follows:

j = the recursive index = 0, 1, ... m-1

p_j = the partial remainder used in the j^{th} cycle

p_o = the dividend

p_m = the remainder

q_j = the j^{th} quotient digit in which the quotient is of the form

$$q_0 \Delta q_1 q_2 \cdots q_m$$

$\uparrow$ radix point

m = the number of digits, radix r, in the quotient

d = the divisor

r = the radix

This relationship and the symbols as defined will be used throughout this report. The relationship is used specifically in the development of range restrictions on the partial remainders in Section 2.3.

Although not germane to the theory of SRT division, it is interesting to note in passing that this relation points to possibilities for accelerating the execution of division. Verbally, the equation says that each partial remainder must be multiplied by the radix (rp_j), i.e. shifted left one digital position and that the selected quotient digit must then be multiplied by the divisor (q_{j+1} d) and subtracted from this shifted partial remainder. The division process will thus be accelerated if the shift and/or the subtraction time is decreased. In practice, all values of q_{j+1} d are stored in registers or are readily available via shift gates from the register containing the divisor. The rapid formation of q_{j+1} d thus reduces to minimizing the necessity for forming awkward multiples requiring an addition, and to accelerating the selection of q_{j+1} d at the divisor input to the adder/subtractor.

Secondly, note that the recursive index, j, is implicitly an inverse function of the radix. When actually implemented on a machine, digits of a higher radix than two are represented by two or more binary bits. A string of ℓ binary digits (bits) is equivalent to $\ell/2$ radix four digits. In general for ℓ bits of radix two, there corresponds $m = \frac{\ell}{\log_2 r}$ digits of radix r, where for practical cases, $r = 2^n$, n = integer > 0. Thus to produce a quotient of given precision, the number of iterations required, and, concomitantly, the execution time is decreased as the radix is increased.

2.2 The Representation of Quotient Digits

As noted in the last section, the use of a higher radix reduces the number of cycles required to perform a division of given precision. The implementation of such a scheme may, however, be costly, and costlier still if quotient digits are represented as they are in manual methods or machine restoring division. In these cases quotient digits have the values 0, 1, 2, ... r-1. With the radix, r, equal four the possible digit values are 0, 1, 2, and 3. A radix four restoring division therefore requires that multiples of 1, 2, and 3 times the divisor be available for subtraction from the partial remainder. The 1 times is of course readily available, the 2 times is formed merely by shifting left one binary position, the 3 times multiple, however, requires extra time and/or hardware. It may be formed by a tripler circuit or by addition of 1 times and 2 times the divisor which is then stored in an auxiliary register. For radix eight, multiples of 3, 5, and 7 times the divisor must be computed and stored.

With SRT division the problem of forming divisor multiples is mitigated by using both plus and minus quotient digit values. The quotient digits are of the form -n, -(n-1), ... -1, 0, 1, ... n, where n is an integer such that $1/2(r-1) \leq n \leq r-1$. Within this range the actual choice of n for a given r is largely a function of design details. The choice is considered further in Section 2.6.

The necessity for the range restriction is as follows: At least r unique digits are required to represent a number, radix r. In the representation introduced above, there are 2n+1 unique digits,

thus the requirement $2n+1 \geq r$. On the other hand, for radix r, the maximum value of a quotient digit, n, should not be greater than the value of the maximum digit representable, thus $n \leq r-1$. Combining these two inequalities yields the restriction stated above.

With plus and minus quotient digits, a higher radix division may be implemented with fewer awkward multiples of the divisor. Now the quotient digits for a radix 4 division are -2, -1, 0, +1, +2. All the necessary multiples of the divisor may be formed by shifting and complementation and require no auxiliary registers.

The second, but probably more significant consequence of this representation of quotient digits is that it introduces redundancy into the representation of the quotient. If $2n > r-1$, then there are more symbols available to represent a number than actually necessary. Some numerical values may therefore be represented in more than one form. For example, with $r = 4$, $n = 2$, and with $\bar{}$ representing negation, the number 6 could be represented as 12, or $2\bar{2}$. As explained in the next sections, this redundancy permits less precision in comparing the divisor and partial remainder in selecting a quotient digit. This statement seems intuitively correct since without redundancy, each quotient digit may be represented only one way and thus must be selected precisely. With redundancy, the quotient digit, thus the comparison of divisor and partial remainder, need not be precise. This non-unique representation does, however, complicate the division in that the redundant form must eventually be converted to a conventional representation.

2.3 Range Restrictions

With the quotient representation now defined, consider the derivation of range restrictions on the partial reminders. Recall from the manual execution of a division that in determining whether a quotient digit is correct or not, one is essentially applying the restriction that $0 \leq p_{j+1} < d$, where p_{j+1} is the result of the subtraction of q_{j+1} times the divisor from the j^{th} partial remainder. If p_{j+1} is not within this range then q_{j+1} is changed until it is. For nonrestoring division, negative partial remainders and negative quotient digits are allowable, thus the range restriction is $|p_{j+1}| \leq |d|$. It seems reasonable, therefore, to hypothesize other division techniques for which $|p_{j+1}| \leq k \mid d \mid$, and which utilize the quotient digit representation introduced in the last section. The upper limit on k will be 1. The lower limit, although not yet obvious, is 1/2, thus $1/2 \leq k \leq 1$.

To show that this is in fact the case, first reconsider the recursive relationship described in Section 2.1 and restated below.

$$p_{j+1} = rp_j - q_{j+1} d \tag{2.3.1}$$

After p_{j+1} is formed on the j^{th} cycle, it is multiplied by the radix r (shifted left); j is increased by one and becomes rp_j of the present cycle. Since $|p_{j+1}| \leq kd$, it follows p_j must obey the same restrictions, i.e.

$$r \mid p_j \mid \leq rk \mid d \mid \tag{2.3.2}$$

Substituting 2.3.1 into 2.3.2 yields

$$-kd \leq rp_j - q_{j+1} \leq kd \tag{2.3.3}$$

At this point the divisor is assumed to be normalized, i.e., restricted to the range $1/2 \leq |d| < 1$. Furthermore, (2.3.1) is normalized with respect to the divisor and rewritten letting $z_j = p_j/d$ and $z_{j+1} = p_{j+1}/d$.

$$z_{j+1} = rz_j - q_{j+1} \tag{2.3.4}$$

Equation (2.3.4) may be interpreted graphically as a plot of z_{j+1} versus rz_j with the quotient digit, q_{j+1} as a parameter. Such a representation shall be called a z - z plot. Recall that the quotient digits assume values -n, -(n-1), ..., -1, 0, +1, ..., n. Figure 2 is such a graph. To facilitate discussion, each plot corresponding to a different quotient digit is called a q-line.

The goal of this section is to demonstrate that a correct division procedure exists which incorporates the above range restrictions and quotient representation. This existence is substantiated if for each value of rz_j in the allowed range there corresponds a quotient digit and a z_{j+1}, also in their allowed ranges. In terms of Figure 2, this means that for any point on the rz_j axis such that $-rk \leq rz_j \leq rk$, one must be able to move on a line segment normal to the rz_j axis and interesect a q-line at a point corresponding to a z_{j+1} within the range $-k \leq z_{j+1} \leq k$. This allowed range is enclosed between the lines $z_{j+1} = k$ and $z_{j+1} = -k$ in Figure 2.

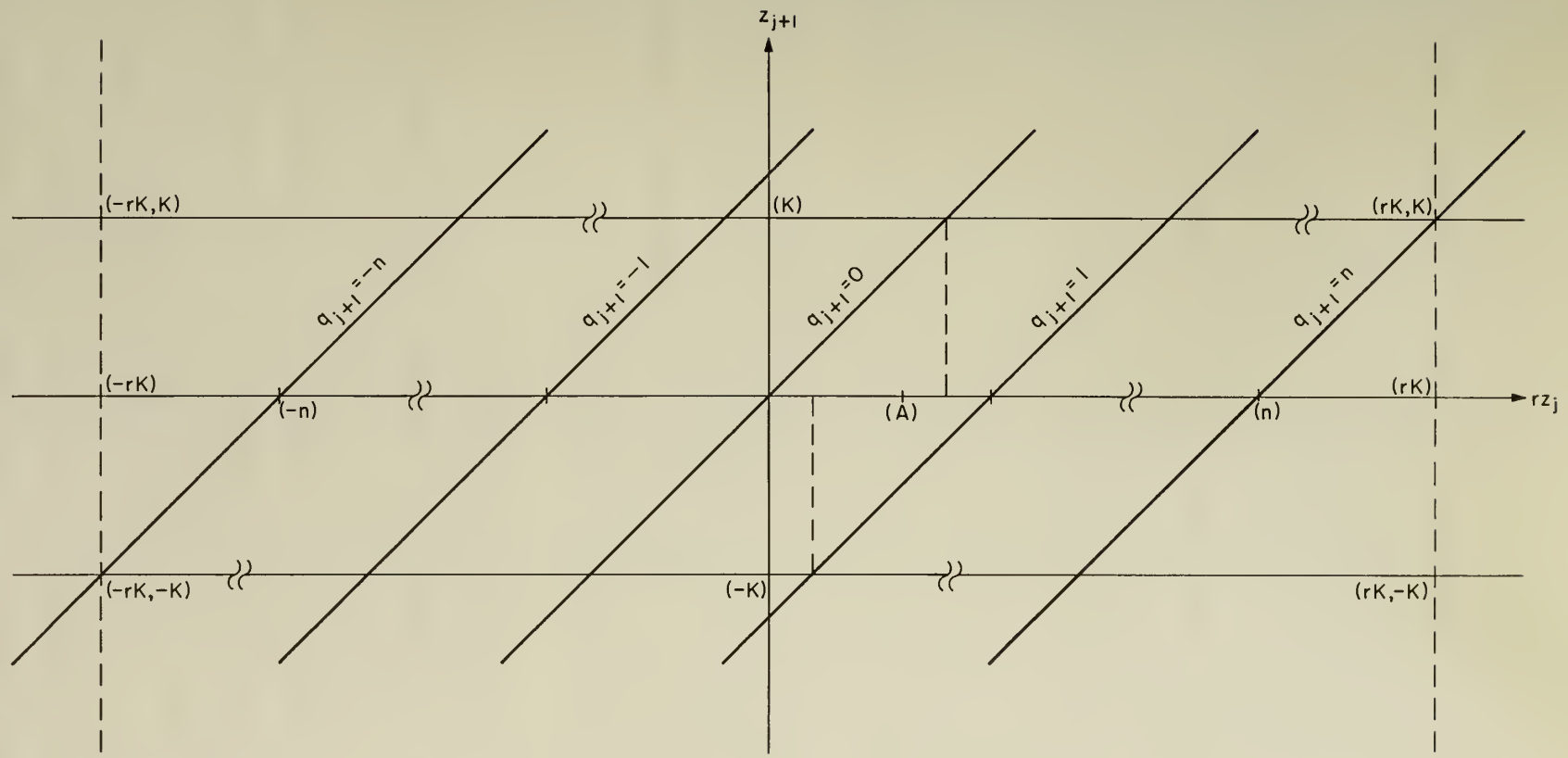


FIGURE 2. Z-Z PLOT OF DIVISION PROCEDURE

To satisfy the foregoing requirements, the maximum value of rz_j, i.e. rk, must occur at the intersection of $z_{j+1} = k$ and the q-line, $z_{j+1} = rz_j - n$. Similarly, the minimum value must occur at the intersection of $z_{j+1} = -k$ and the q-line, $z_{j+1} = -rz_j + n$. These bounds on rz_j are indicated by the dashed vertical lines of Figure 2.

Figure 2 now points to the value of k in terms of r and n. At the upper right vertex of the bounding rectangle, $z_{j+1} = k = rz_j - n$. But since $rz_j = rk$,

$$k = \frac{n}{r-1} \tag{2.3.5}$$

The division is now characterized by tangible parameters, namely the radix and the maximum value of quotient digits. Combining (2.3.5) with the restriction on n, $\frac{r-1}{2} \leq n \leq r-1$, verifies the statement at the beginning of this section, $1/2 \leq k \leq 1$.

2.4 Redundancy in the Quotient Representation

Section 2.2 indicated that the quotient digit representation of SRT division introduces redundancy into the quotient. This fact is also manifested in Figure 2 in the regions on the rz_j axis for which either one of two q-lines may be legitimately selected. For example, at point A one may move vertically upward to the $q_{j+1} = 0$ line or downward to the $q_{j+1} = +1$ line. In either case the quotient digit is correct. Figure 3, a specific case of Figure 2, testifies to the fact that this freedom of choice is not merely the result of an inaccurately drawn graph. Here r = 4, n = 2. The vertical dashed lines define the overlap regions.

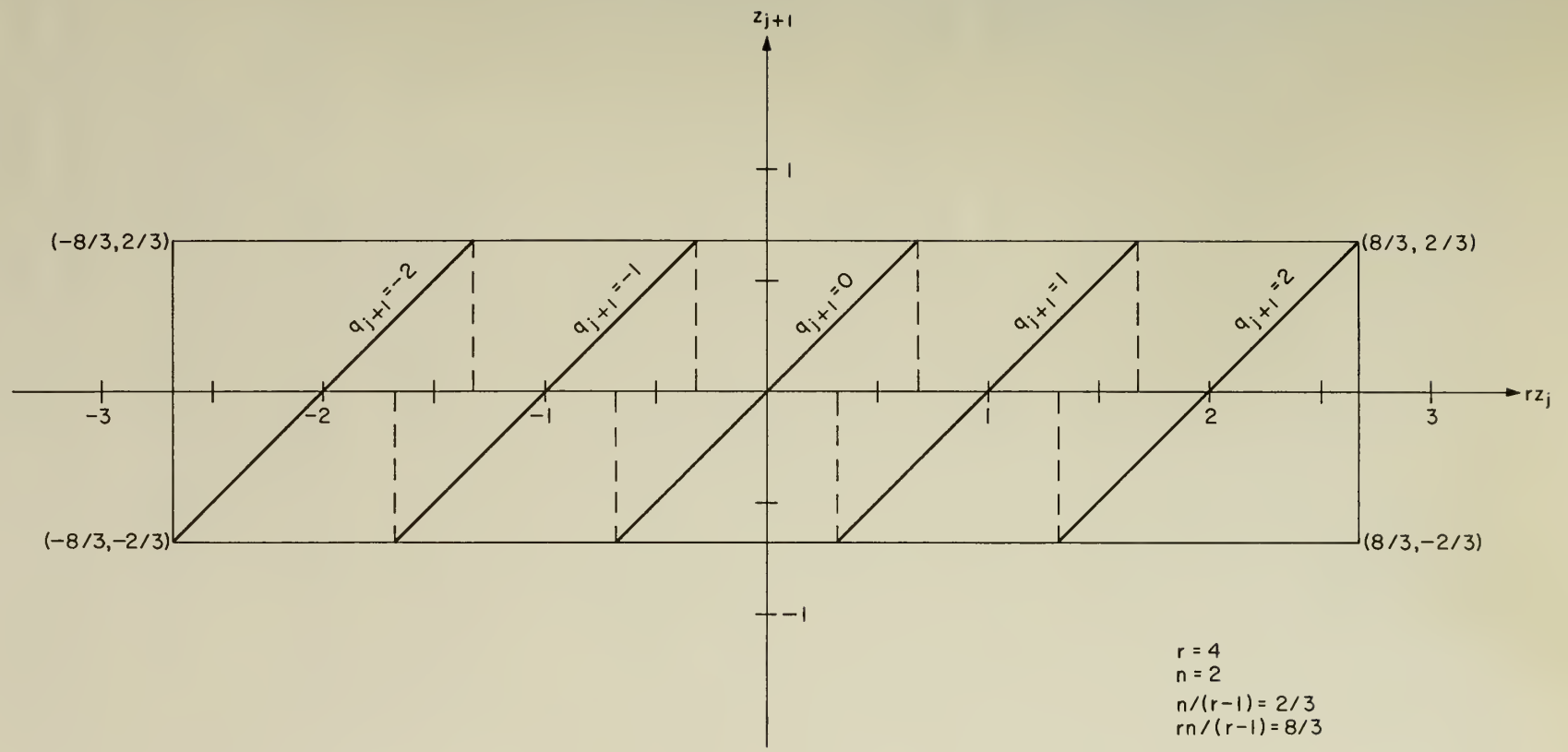


FIGURE 3. Z–Z PLOT WITH r = 4, n = 2

The production of a redundant quotient requires extra hardware and perhaps time, to convert it to a conventional binary representation acceptable by programmers and other sections of a machine. This conversion is discussed at greater length in Section 2.7. The conclusion of the section is that the positive consequences of a freedom in quotient digit selection overshadow the cost of conversion. With no redundancy, the divisor and the shifted partial remainder must be compared (usually by subtraction) to the full precision defined for the machine. With redundancy, the designer is at liberty to inspect fewer bits of the divisor and shifted partial remainder than define full precision. Handling fewer bits may save time and hardware: these ramifications are explored further in the chapter concerning implementation. In Figure 3, for example, a correct quotient digit is selected knowing $rz_j = \frac{rp_j}{d}$ to a precision only great enough to contain it within an overlap region. Exactly what precision is required for a given value of r and n is the subject of the next section.

In terms of z - z plots such as Figures 2 and 3, the redundancy is proportional to the width of the overlap regions. The width of this region in terms of n and r is found as follows: Consider two adjacent lines of Figure 2, i.e., $z_{j+1} = rz_j - i$ and $z'_{j+1} = rz_j - (i-1)$. The overlap, Δrz_j, is the difference between rz_j for $z_{j+1} = \frac{n}{r-1}$ and rz_j for $z'_{j+1} = \frac{-n}{r-1}$. Solving for this difference yields $\Delta rz_j = \frac{2n}{r-1} + 1$. The ratio $\frac{n}{r-1}$ is therefore a measure of redundancy.

As redundancy (width of overlap region) is increased, the required precision of inspection of divisor and partial remainder, and thus hopefully the execution time, is decreased. It, therefore, appears that for a given r, n should be as large as possible, i.e., n should equal r-1. Such a choice may not be practical, however, since n = h, requires the ability to form h multiples of the divisor. The choice of n is therefore bound up in the usual trade off between time and hardware.

2.5 The P-D Plot

Now consider another graphical representation of the division procedure. This construction, suggested by C. V. Freiman of the IBM Corporation [5] is useful in further describing SRT division and in computing the required precision of inspection of the divisor and shifted partial remainder. The basis for the plot is the recursive relationship

$$p_{j+1} = rp_j - q_{j+1} d \qquad (2.1.1)$$

as described in Section 2.1 together with the range restriction

$$\left| p_{j+1} \right| \leq \frac{n}{r-1} d$$

developed in Section 2.3. The figure is thus essentially a plot of partial remainder versus divisor values and therefore in this report shall be referred to as a P-D plot.

Solving the recursive relationship for rp_j yields

$$rp_j = p_{j+1} + q_{j+1}\, d. \tag{2.5.1}$$

For a fixed quotient digit, the upper limit of rp_j as a function of the divisor, d occurs when p_{j+1} is maximum, i.e. when

$$p_{j+1} = \frac{n}{r-1}\, d,$$

thus

$$rp_{j\ max} = \left(\frac{n}{r-1} + q_{j+1}\right) d. \tag{2.5.2}$$

Likewise, the lower limit occurs with $p_{j+1} = \frac{-n}{r-1}\, d$, thus

$$rp_{j\ min} = \left(\frac{-n}{r-1} + q_{j+1}\right) d. \tag{2.5.3}$$

These linear equations may be plotted as functions of d with q_{j+1} as a parameter ranging from -n to +n in steps of 1. The area between $rp_{j\ max}$ and $rp_{j\ min}$ for a given $q_{j+1} = i$ will be denoted the <u>q(i) area</u>.

The division procedure is now determined. A given value of divisor, d and the j^{th} shifted partial remainder will specify a point in a q(i) area. The digit i will be the value of the next quotient digit q_{j+1} which in turn is used in forming the next partial remainder.

In this representation the redundancy is manifested as overlapping of the q(i) regions, i.e. some pairs of d and rp_j will specify a point for which either $q_{j+1} = i$ or $q_{j+1} = i - 1$ is a valid choice.

Figure 4 is an example of a P-D plot for a division with r = 4, n = 2. The equations for the lines plotted, 2', 2, etc., are given in Table 1. The region for which $q_{j+1} = 2$ is a valid choice, i.e. the q(2) area, is between lines 2' and 2; the q(1) area is between lines 1' and 1, and so forth. Note the overlap between q(i) areas, for example, the region between line 1' and 2 in which either the choice $q_{j+1} = 1$ or $q_{j+1} = 2$ is correct. Note further that the figure is symmetric about both axes.

On the right half of Figure 4 (the same may be done on the left), "steps" have been drawn within the overlap of the q(i) regions. The width of a "tread" (constant rp_j, d varying) defines a <u>divisor interval</u>, the value of rp_j for each tread defines a comparison constant, the distance between comparison constants defines a <u>partial remainder interval</u>. Phrased in this terminology, division consists of locating a given divisor value within the appropriate divisor interval, locating the shifted partial remainder within the appropriate interval (using comparison constants), and selecting a value of q_{j+1} enclosed by the intersection of the boundaries of these intervals. Since a divisor and partial remainder must be located only to within an <u>interval</u>, they need not be inspected to full precision in selecting a correct quotient digit. Here is where the redundancy pays dividends.

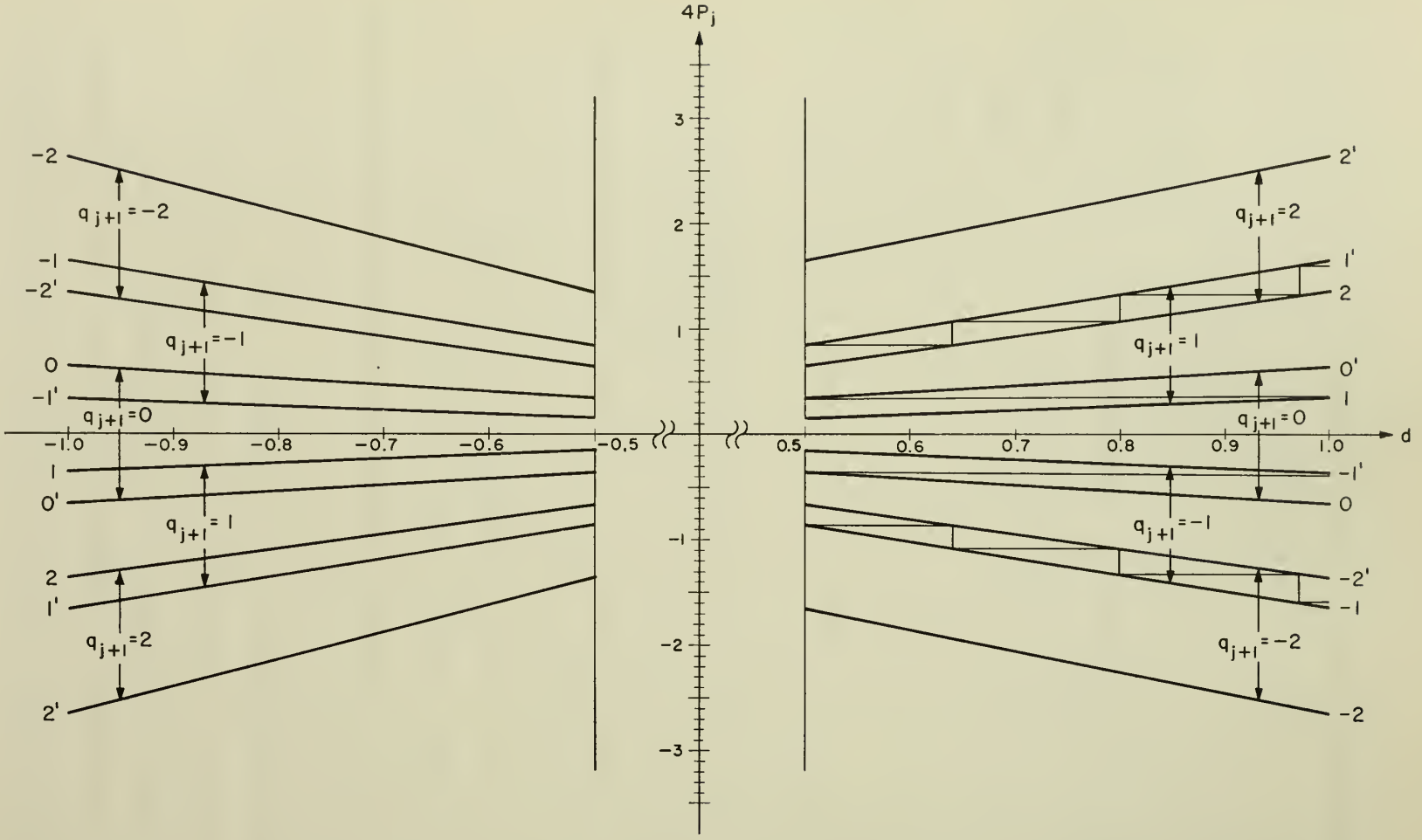


FIGURE 4. P-D PLOT WITH r=4, n=2

$$rp_j = \pm \frac{n}{r-1} d + q_{j+1} d$$

$$r = 4$$

Designation in Figure 3	q_{j+1}	p_{j+1}	Equation rp_j =
2'	2	2/3 d	8/3 d
2	2	-2/3 d	4/3 d
1'	1	2/3 d	5/3 d
1	1	-2/3 d	1/3 d
0'	0	2/3 d	2/3 d
0	0	-2/3 d	-2/3 d
$\bar{1}'$	$\bar{1}$	2/3 d	-1/3 d
$\bar{1}$	1	-2/3 d	-5/3 d
$\bar{2}'$	$\bar{2}$	2/3 d	-4/3 d
$\bar{2}$	$\bar{2}$	-2/3 d	-8/3 d

Table 1. Equations Defining the Regions of Figure 4.

Techniques for selecting divisor intervals and comparison constants are detailed in the next two sections. At this point, however, we shall make several general observations. First, as we shall soon discover, the comparison constants are compared with the high order N_p bits of the shifted partial remainder and, similarly, the end points of the divisor intervals are compared with the N_d high order bits of the divisor. The comparison constants and end point of the divisor intervals should therefore be numbers which are representable with N_p and N_d bits, respectively. The choices illustrated in Figure 4 which maximized the width of the divisor intervals do not meet this requirement.

In Figure 5, however, more practical choices are shown. The dashed lines represent the theoretical choices used in Figure 4. Now, although the number of steps has been increased, the boundaries fall at points easily representable in binary notation. Note that inspection of 4 bits plus sign of the partial remainder and divisor is sufficient to locate the correct choice of quotient digit.

The second observation is that the choice of divisor intervals and comparison constants is bound up with the required precision of inspection of the partial remainder and divisor; if, for example, the divisor intervals widths are increased, the required precision of divisor inspection, (number of bits) may be decreased. Furthermore, the maximum precision of inspection of the divisor is determined by the divisor interval of smallest width. By inspection of Figure 5, the reader might guess where this step is, but, we shall now locate it analytically. The result of this derivation will be useful in the next sections.

The length of a divisor interval is limited by the boundaries of the overlap region. The maximum precision of inspection is required where the divisor interval is minimum. To determine where this minimum divisor interval occurs consider the detail of the overlap of the q(i) and q(i-1) regions shown in Figure 6.

For a given value of rp_j, the maximum width of a divisor interval is

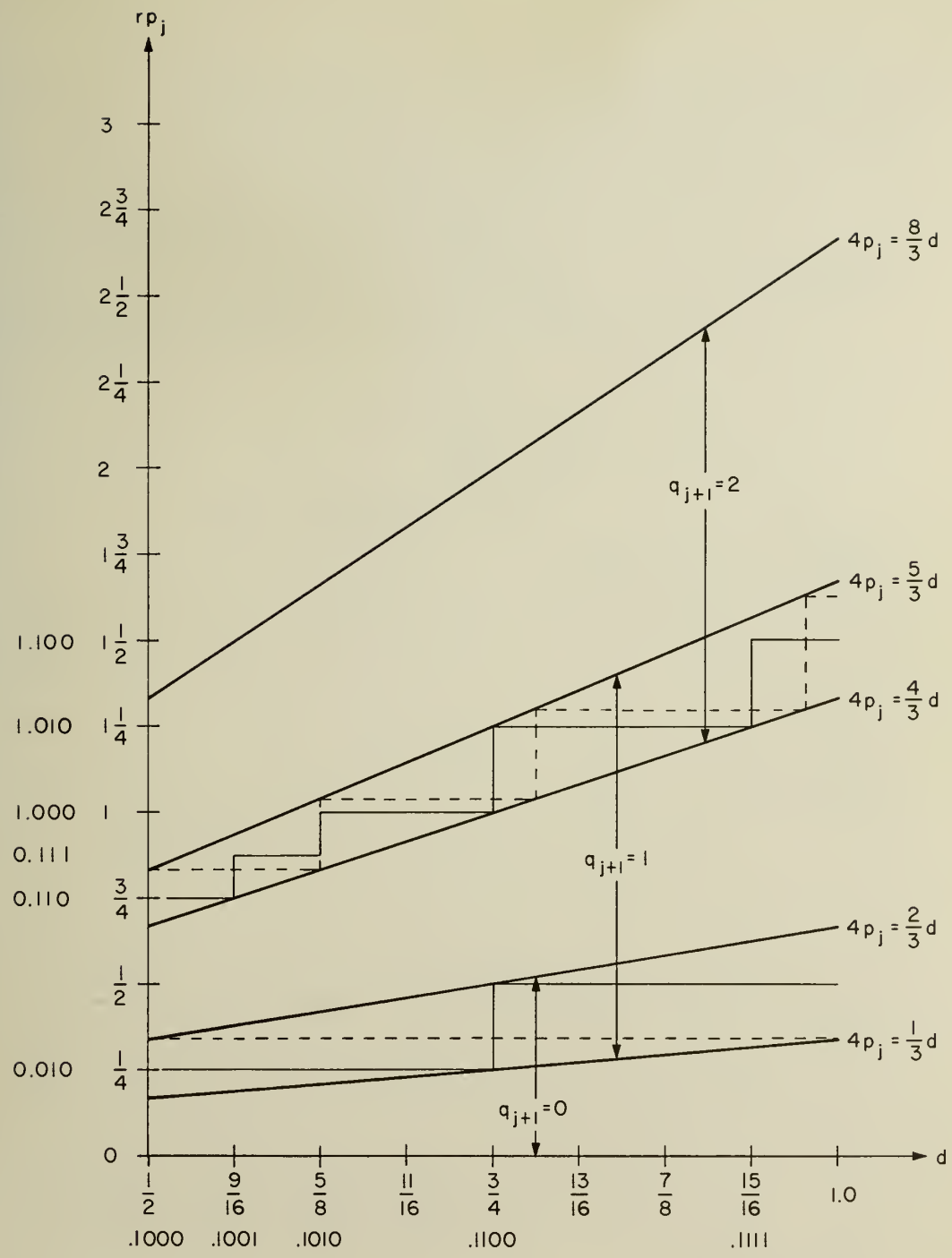


FIGURE 5. DIVISOR INTERVALS AND COMPARISON CONSTANTS WITH r=4, n=2

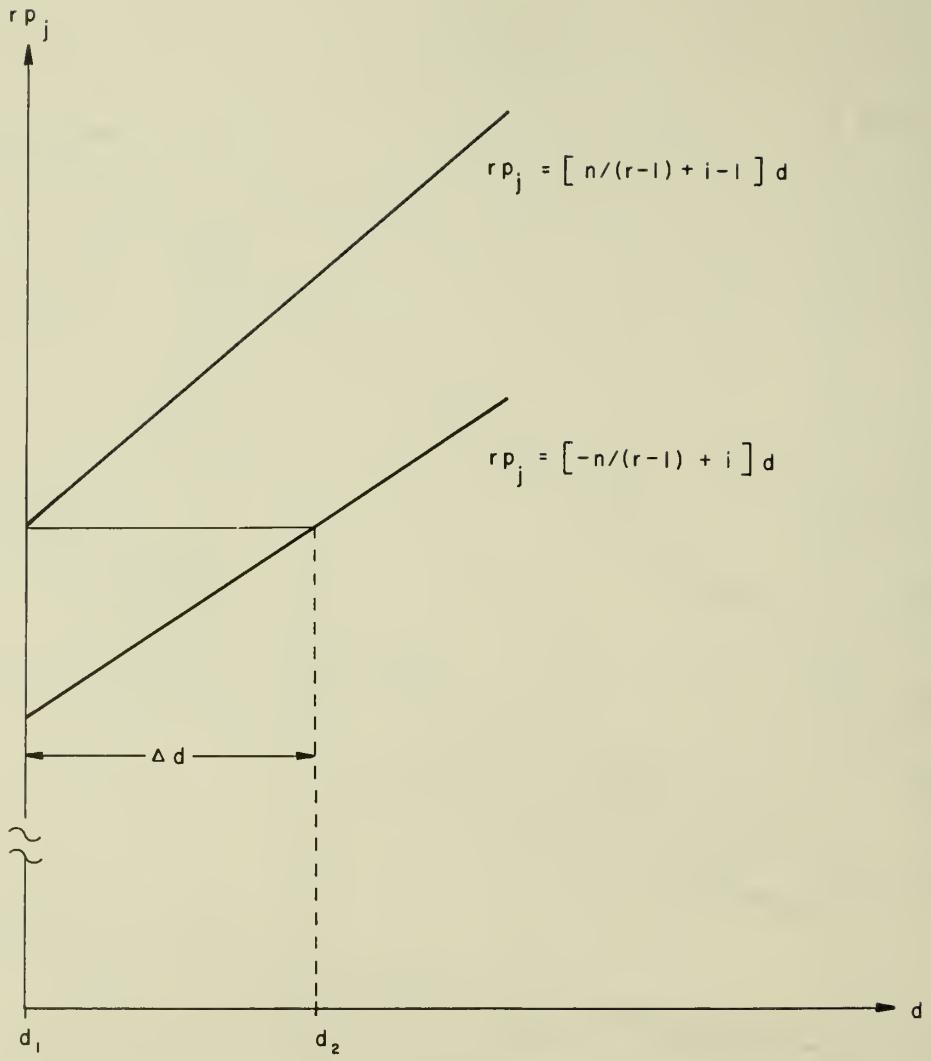


FIGURE 6. DETAIL OF A P-D PLOT OVERLAP REGION

$$\Delta d = d_2 - d_1 = \frac{rp_j}{\frac{-n}{r-1} + i} - \frac{rp_j}{\frac{n}{r-1} + i-1}$$

$$= rp_j R \quad \frac{2n - R}{R^2 i^2 - R^2 i + nR - n^2} \qquad (2.5.4)$$

where $R = (r-1)$.

The interval Δd is minimum, when i is maximum and rp_j is minimum. The maximum value of i is n, the minimum value of rp_j for $q_{j+1} = n$ will occur when the upper bound of the overlap region intersects $d = 1/2$, i.e., when $d_1 = 1/2$. The precision of required inspection of divisor is thus determined by the divisor interval closest to $d = 1/2$ and between $q_{j+1} = n$ and $q_{j+1} = n-1$.

Robertson has introduced the selection ratio, [5] which is defined as the ratio of the slope of the lower bound of an overlap region to the slope of the upper bound. This ratio is a relative measure of the width of the divisor interval for which a single comparison constant is valid. From Figure 6 , it appears that σ_i, (the selection ratio between $q_{j+1} = i$ and $q_{j+1} = i-1$) is

$$\sigma_i = \frac{i(r-1) - n}{(i-1)(r-1)+n} \qquad (2.5.5)$$

The difficulty of selection is proportional to σ_i and as indicated earlier is most difficult for $i = n$.

The selection ratio may be used to compute another parameter the minimum number of divisor intervals necessary to span a given overlap region. If $a \leq d \leq b$, then this minimum is the smallest

integer, S_i such that $\sigma_i^{S_i} \leq \frac{a}{b}$, i.e., S_i = integer part of $\frac{\log a - \log b}{\log \sigma_i}$. For example with $1/2 \leq d \leq 1$, $i = n = 2$, $r = 4$, $\sigma_i = .8$ and $S_i = 4$. Note that this agrees with the graphical results of Figure 4. The number of steps between line 1' and line 2 is four.

2.6 The Cost of Quotient Digit Selection

2.6.1 General

To this point we have established that an important feature of SRT division is the ability to select quotient digits from truncated versions of the divisor and shifted partial remainder. We now turn to the more specific question of what precision is required in these approximations, i.e., how many bits of the divisor and shifted partial remainder must be inspected to guarantee correct quotient digit selection. In a sense, this required precision is the cost of quotient digit selection.

The cost will be shown to be a function of the choice of radix and to a certain extent, of the method of selecting the quotient digits. Robertson [5] has suggested that the mechanism for selection of quotient digits may be viewed as a limited precision model of the full precision division. This concept is exemplified in the following example.

A radix 256 division would require eight quotient bits per shift of partial remainder. To generate these eight bits, as shown in Section 2.6.2, 12 bits of the partial remainder and 13 bits of the divisor are presented to a division mechanism which need be only elaborate enough to produce eight bits of quotient from a 12 bit

dividend and a 13 bit divisor. The results of this limited precision division (eight bits) are returned to the full precision mechanism as part of the full precision quotient and are used in forming the next full precision partial remainder. Note that the number defining full precision may be changed in discrete steps by changing the number of "calls" to the model division. Furthermore, the model division scheme may be quite different from that of the full precision division.

For purposes of computing costs of quotient selection, we shall consider two classes of model division procedures. The first will be those involving the use of an auxilary arithmetic unit and employing addition and/or subtraction in forming the quotient digits. Examples of schemes in this class include a radix four SRT division performed in the exponent arithmetic unit or the procedure suggested by Wallace [9] which is logically equivalent to forming the approximate reciprocal of the divisor and multiplying by the partial remainder. This class will be referred to as arithmetic models.

The second class consists of those methods which are the logical equivalent of a table look-up. This technique may be viewed as the direct implementation of a P-D plot, i.e., decoding the divisor interval, the partial remainder interval and producing the quotient digit indicated by their intersection. This class will be referred to as table look-up models.

Before considering these two type models in further detail, let us state more precisely the conditions which must be obtained in

the choice of model division and precision of inspection. Let

m = the number of bits to the right of the radix point of divisor and dividend.

$\hat{rp}_j$ = the truncated version of the shifted partial remainder.

ϵ = the number of bits to the right of the radix point in $\hat{rp}_j$.

$\Delta p = \pm (2^{-\epsilon} - 2^{-m}) \approx \pm 2^{-\epsilon}$, the uncertainty in rp_j.

$\hat{d}$ = the truncated version of the divisor.

δ = the number of bits to the right of the radix point in $\hat{d}$.

$\Delta d = \pm (2^{-\delta} - 2^{-m}) \approx \pm 2^{-\delta}$, the uncertainty in $\hat{d}$.

The following cost criterion summarizes the requirements on the quotient selection mechanism, Δd and Δp.

Cost criterion: Given the approximations $\hat{rp}_j \pm \Delta p$ and $\hat{d} \pm \Delta d$, the integer result of $\hat{rp}_j / \hat{d} = i$ performed in the model must be such that on the appropriate P-D plot, the rectangle defined by $(\hat{d} \pm \Delta d,\ rp_j \pm \Delta p)$ is entirely within the $q(i)$ region.

2.6.2 Cost Determination for an Arithmetic Model

We first consider the determination of the cost for a division using an arithmetic model. In this case $\hat{rp}_j$ and $\hat{d}$ are presented to a limited precision arithmetic unit and the division carried out to produce a rounded integer quotient. If the bit position to the right of the radix point in the model is "1", the integer

portion is increased by one and truncated, otherwise the result is merely truncated. This rounding is necessary if the cost criterion is to hold for an arithmetic model.

Equation 2.5.4 indicated that maximum precision is required in the overlap of the q(n) and q(n-1) regions in the vicinity of d = 1/2. The precision determined here will be sufficient for any other region of the P-D plot. Figure 7 is a detail of this region.

Two additional factors must now be considered: a redundantly represented partial remainder and a negative divisor. As illustrated in the next chapter, a division scheme which meshes well with multiplication must cope with redundantly represented partial remainders. One consequence of the representation is that the truncation error (Δp) attributable to considering only a few higher order bits of the partial remainder may be either positive or negative. When a negative (2's complement) divisor is permitted, truncation error may also be negative.

In the divisor interval $1/2 \pm \Delta d$, the dividing line between the selection of q = n and q = n-1 is $\hat{rp}_j = 1/2(n - 1/2)$ since $\hat{rp}_j/\hat{d} = 2 \times 1/2(n - 1/2) = n - 1/2$ which must be rounded to n. For the cost criterion to hold, the rectangle $(1/2 \pm \Delta d, 1/2(n - 1/2) \pm \Delta p)$ must not extend below the bottom of the overlap region defined by $rp_j = (n - 2/3)d$. Such a rectangle is indicated by the dashed lines in Figure 7. Since this rectangle is not unique, there is some available trade off between Δp and Δd. To achieve more quantitative

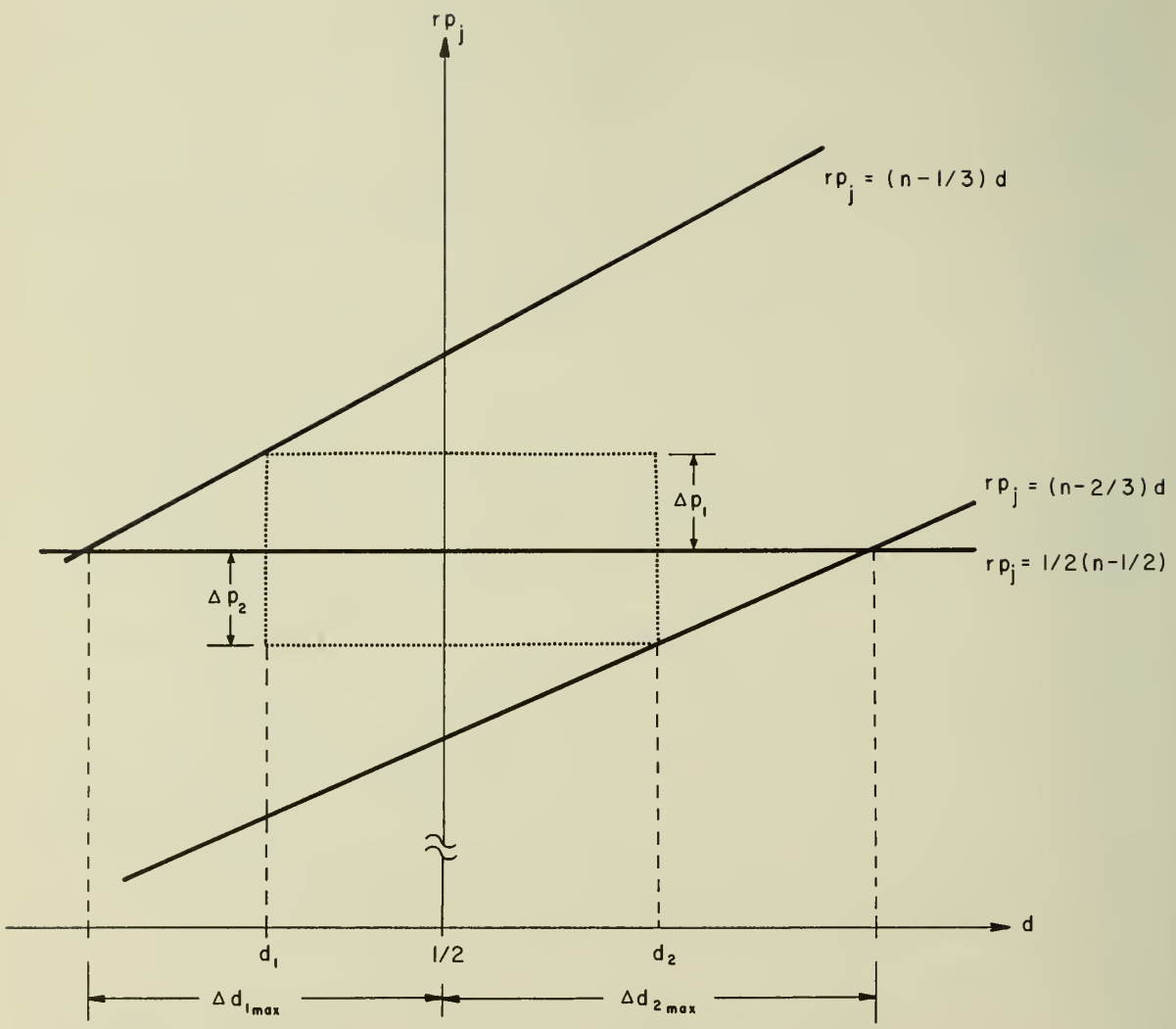


FIGURE 7. COST CALCULATION FROM P-D PLOT

results, we now limit the analysis to a special but useful case: that in which the radix is of the form $r = 2^{2k}$; where k is a positive (non-zero) integer.

A division with $r = 2^{2k}$ may be implemented with a cascade of k adder/subtractors with multiples of 1 times and 2 times the divisor available to the first stage of the cascade, 4 times and 8 times to the second, and so forth through $2^{(2k-2)}$ times and $2^{(2k-1)}$ times available to the k^{th} stage. In this case, n, the largest multiple of the divisor which may be formed, is the sum of the largest multiple which may be formed at each stage in the cascade, i.e. $n = 2 + 8 \ldots + 2^{(2k-1)}$. Furthermore, the sum of this geometric series is $\frac{n}{r-1} = 2/3$. Thus we shall consider the case $r = 2^{2k}$, $n = 2/3(r-1)$.

For practical implementation, the rectangular region defined horizontally by Δp will be symmetric about $d = 1/2$ and $rp_j = 1/2(n-1/2)$. Referring to Figure 7, note that Δd must be smaller than the smaller of $\Delta d_{1\ max}$ and $\Delta d_{2\ max}$. The following demonstrates that $\Delta d_2 < \Delta d_{1\ max}$.

$$\Delta d_{2\ max} = 1/2 \left(\frac{n - 1/2}{n - 2/3} - 1 \right) \qquad (2.6.1)$$

$$\Delta d_{1\ max} = 1/2 \left(\frac{-n - 1/2}{n - 1/3} + 1 \right)$$

$$\Delta d_{1\ max} - \Delta d_{2\ max} = 1 - \frac{n^2 - n + 1/4}{n^2 - n + 2/9} \qquad (2.6.2)$$

Since

$$\frac{n^2 - n + 1/4}{n^2 - n + 2/9} > 1$$

$$\Delta d_{1\ max} - \Delta d_{2\ max} < 0$$

$$\Delta d_{1\ max} < \Delta d_{2\ max} \tag{2.6.3}$$

Thus choosing $\Delta d \leq \Delta d_{1\ max}$ will insure that the rectangle will fit horizontally.

Similarly

$$\Delta p_1 = (n - 1/3)d_1 - 1/2(n - 1/2) \tag{2.6.4}$$

$$\Delta p_2 = - (n - 2/3)d_2 + 1/2(n - 1/2)$$

$$\Delta p_1 - \Delta p_2 = (n - 1/3)d_1 + (n - 2/3)d_2 - (n - 1/2) \tag{2.6.5}$$

let

$$d_1 = 1/2 - \Delta d$$

$$d_2 = 1/2 + \Delta d \tag{2.6.6}$$

Substituting (2.6.6) into (2.6.5) yields

$$\Delta p_1 - \Delta p_2 = \frac{-\Delta d}{3} \leq 0$$

thus

$$\Delta p_1 \leq \Delta p_2 \tag{2.6.7}$$

As implied earlier, if we are certain that $\hat{rp}_j = 1/2(n - 1/2)$ will produce the quotient selection, $q_{j+1} = n$, then $\Delta p \leq \Delta p_2$ will be sufficient. If we cannot guarantee this, then $\Delta p \leq \Delta p_1$ must hold.

We shall adopt the latter, more cautious approach. If we selected the former, then the $(n - 1/3)$ term in equation 2.6.13 would be replaced by $(n - 2/3)$. The results in Table 2, however, will be the same.

Recalling that $\Delta d = 2^{-\delta}$ we want

$$2^{-\delta} \leq \Delta d_{1\ max} \tag{2.6.8}$$

which from 2.6.1 becomes

$$2^{-\delta} \leq 1/2 \left(\frac{n - 1/2}{n - 1/3} - 1 \right) \tag{2.6.9}$$

where

$$n = 2/3(2^{2k} - 1)$$

Let

$I(x) = x$ if x is an integer.

= next larger integer if x is not an integer.

The minimum value of δ is therefore

$$\delta_{min} = -I\left[\log_2\left(1/2\left(1 - \frac{n - 1/2}{n - 1/3}\right)\right)\right] \qquad (2.6.10)$$

Possible values of δ are thus

$$\delta = \delta_{min}, \delta_{min} + 1, \ldots m \quad . \qquad (2.6.11)$$

Similarly since $\Delta p = 2^{-\epsilon}$, combining 2.6.7 and 2.6.4 yields

$$2^{-\epsilon} \leq 1/12 - 2^{-\delta}(n - 1/3) \qquad (2.6.12)$$

and thus

$$\epsilon = -I\left[\log_2\left(1/12 - 2^{-\delta}(n - 1/3)\right)\right] \qquad (2.6.13)$$

where δ is defined by 2.6.11.

Now let

N_d = number of bits of $\hat{d} = \delta$.

N_p = number of bits of $\hat{rp}_j = \epsilon + 2k$

Note also that the sign of d and rp_j must be known to model. Table 2 summarizes the results of equations 2.6.11 and 2.6.13 for k = 1, 2, 3, 4. Note that ϵ approaches a lower limit of 4 when the 1/12 term in 2.6.13 becomes dominant.

k	r	n		δ	ϵ	N_d	N_p
1	4	2	δ_{min} =	5	5	5	7
				6	5	6	7
				7	4	7	6
				8	4	8	6
				⋮	⋮	⋮	⋮
				m	4	m	6
2	17	10	δ_{min} =	7	7	7	11
				8	5	8	9
				9	4	9	8
				10	4	10	8
				⋮	⋮	⋮	⋮
				m	4	m	8
3	64	42	δ_{min} =	9	9	9	15
				10	5	10	11
				11	4	11	10
				12	4	12	10
				⋮	⋮	⋮	⋮
				m	4	m	10
4	256	170	δ_{min} =	11	11	11	19
				12	5	12	13
				13	4	13	12
				14	4	14	12
				⋮	⋮	⋮	⋮
				m	4	m	12

Table 2. Costs for Arithmetic Models

Thus it appears there are three feasible cases for which the cost of inspection is as follows:

Case 1

$$N_p = 4k + 3$$

$$N_d = 2k + 3$$

Case 2

$$N_p = 2k + 5$$

$$N_d = 2k + 4$$

Case 3

$$N_p = 2k + 4$$

$$N_d = 2k + 5$$

Case three would probably be the most practical case to implement since N_p is minimum. N_p bits of the redundantly represented partial remainder must be converted into conventional form before each model division. Since this assimilation is essentially a serial process, the assimilation time is directly proportional to N_p.

2.6.3 Cost Determination for a Table Look-Up Model

This class of model is a logical implementation of the P-D diagram. In its most brute force form, this model may be viewed as a grid or matrix with vertical lines which are the outputs of decoders applied to $\hat{d}$ and with the horizontal lines which are the outputs of the decoders applied to $\hat{rp}_j$. At each intersection of the lines is and AND gate with one input connected to the vertical line, the other to the horizontal line. Each point of intersection corresponds to a

quotient digit value, i, and thus the output of each AND gate is connected to the input of the appropriate ØR gate the true output of which is $q_{j+1} = i$.

The overlap regions are divided by steps as discussed in Section 2.5 such that the cost criterion (Section 2.6.1) will hold in all intervals. To determine the required N_p and N_d in this case, we again consider the worst case region of the P-D plot where d = 1/2 and between q(n) and q(n-1) as shown in Figure 7.

Again, if we choose the dividing line between $q_{j+1} = n$ and $q_{j+1} = n-1$ to be at 1/2(n - 1/2), then the calculations of Section 2.6.2 also hold for the table look-up case with $r = 2^{2k}$. Recall, however, that we generally wish to minimize N_p since this will reduce the assimilation time in forming $\hat{rp}_j$ in each cycle. We can accomplish this by selecting the comparison constants, the dividing line between choice of quotient digit values, as close to the top of an overlap region as possible.

In the arithmetic models, the comparison constants are implicit in the model, and thus, for example, we had no choice but to use 1/2(n - 1/2) in the cost calculations. In the present case, however, we may select any value which is within the overlap region and an integer multiple of $2^{-\epsilon}$.

The value of 1/2(n - 1/2) is always an exact binary number, specifically a number with a fractional part of 3/4. The distance from 1/2(n - 1/2) to the upper limit of the overlap region along d = 1/2 is 1/2(n - 1/3) - 1/2(n - 1/2) = 1/12. This means that the

largest comparison constant we may choose in this region without increasing ϵ to be greater than four is $1/2(n - 1/2) + 1/16$. If we design the logic such that $\hat{rp}_j = 1/2(n - 1/2) + 1/16$ and $\hat{d} = 1/2$ selects $q_{j+1} = n$, then Δd and Δp cost calculations are as follows: In this case

$$2^{-\delta} \leq \Delta d_{max}$$

$$2^{-\delta} \leq 7/48 \cdot \frac{1}{n - 2/3}$$

$$2^{-\epsilon} \leq 7/48 - 2^{-\delta}(n - 2/3)$$

In the same manner as that outlined in the last section we obtain Table 3 and the three cases.

Case 1

$$N_p = 2k + 4$$

$$N_d = 2k + 3$$

Case 2

$$N_p = 2k + 4$$

$$N_d = 2k + 4$$

Case 3

$$N_p = 2k + 3$$

$$N_d = 2k + 5$$

The first entry $N_d = 4$, $N_p = 6$ is not included in the above linear equations but this is the most practical case for k = 1, radix

k	n		δ	ϵ	N_d	N_p
1	2	δ_{min} =	4	4	4	6
			5	4	4	6
			6	3	4	6
			7	3	3	5
			⋮	⋮	⋮	⋮
			m	3	m	5
2	10	δ_{min} =	7	4	7	8
			8	4	8	8
			9	3	9	7
			⋮	⋮	⋮	⋮
			m	3	m	7
3	42	δ_{min} =	9	4	9	10
			10	4	10	10
			11	3	11	9
			⋮	⋮	⋮	⋮
			m	3	m	9
4	170	δ_{min} =	11	4	11	12
			12	4	12	12
			13	3	13	11
			⋮	⋮	⋮	⋮
			m	3	m	11

Table 3. Costs for Table Look-Up Models

four. By comparison with the results of Section 2.6.2, note that for a given k, a case may be found for which a table look-up model requires fewer bits of comparison than the corresponding arithmetic model.

2.7 Quotient Conversion

The quotient developed by SRT division will in general include negative digits and eventually must be converted to a conventional binary form. This conversion time and hardware is the greater part of the price paid for the accrued advantages of redundancy.

First consider a specific case: conversion of a result produced by a non-restoring division. Here quotient representation is the same as that discussed in Section 2.2 except that zero is not an allowable digit. The algorithm for such a conversion is illustrated in Figure 8. This conversion may be performed sequentially as the quotient digits are generated, and thus requires no additional terminal operations. The digit q_{j+1} is unchanged if it is positive, otherwise it is replaced by $r + q_{j+1}$, and the adjacent higher order digit q_j, decreased by 1. Note that since zero is not a permissible digit, there is no requirement for a borrow propagation in decreasing q_j by 1. The hardware required is of the order of a two digit subtractor.

It is not generally possible, however, to perform SRT division not allowing $q = 0$. Non-restoring division may be viewed as SRT division with $n = r-1$. For this case, the $q(0)$ region of a P-D plot is completely overlapped by the $q(1)$ and $q(-1)$ regions. The quotient digit value $q = 0$ may, therefore, be eliminated and the conversion consequently simplified to that of Figure 8. For cases of SRT division with $n < r-1$, the $q(0)$ region is not subsumed by other regions and thus $q = 0$ must be allowed if the division is to be completely defined.

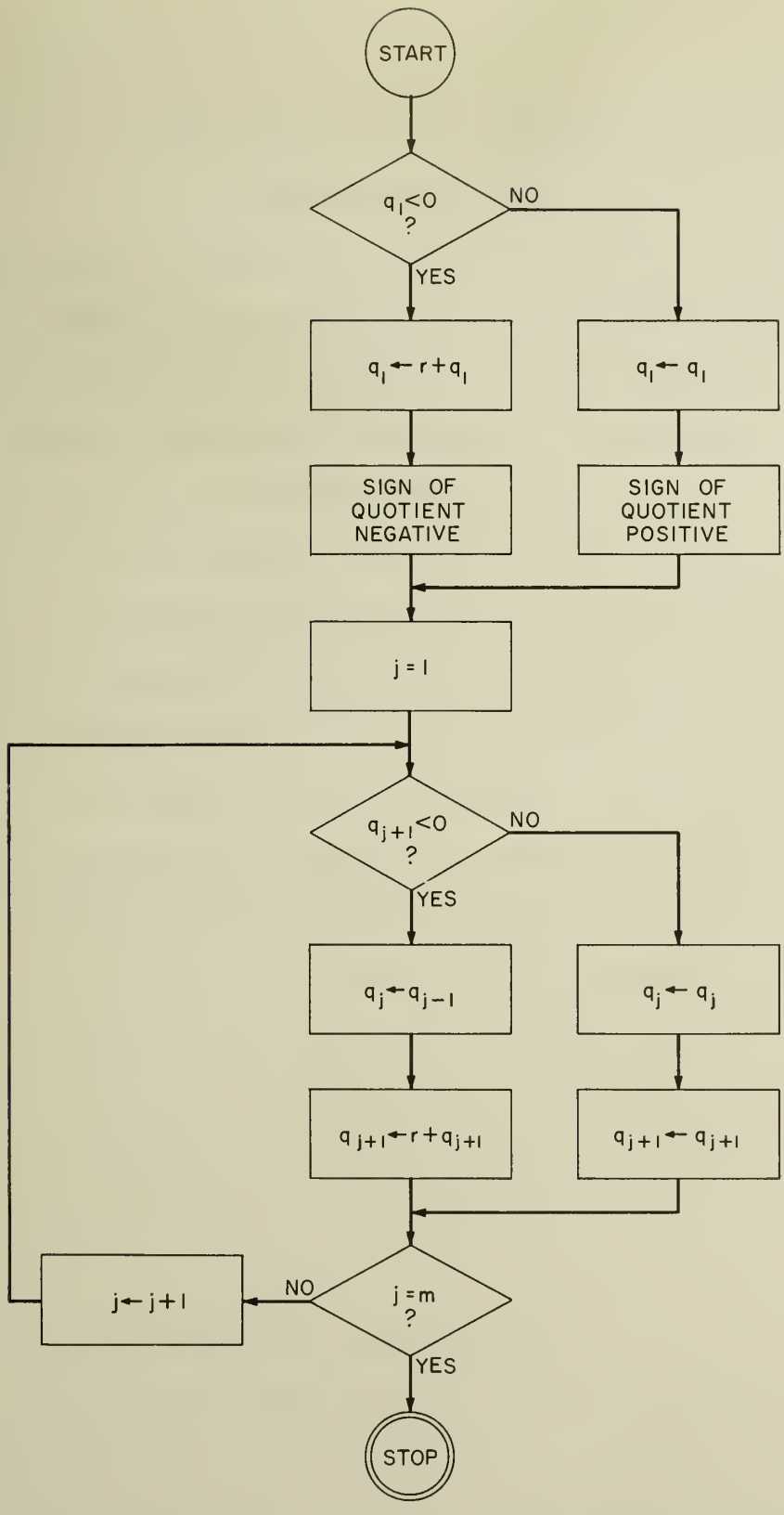


FIGURE 8. QUOTIENT CONVERSION FOR NON-RESTORING DIVISION

With the possibility of $q = 0$, the conversion is complicated: the algorithm of Figure 8 is no longer adequate, for now the difference $q_j - 1$ may require a borrow from q_{j-1}. Furthermore, this borrow must propagate to the left until it encounters a non-zero digit. This potential for borrow propagation requires that the equivalent of a full precision subtractor be available to the quotient register if conversion is to occur as the quotient digits are generated.

Alternately, the full precision quotient may be generated and stored in the redundant form and then converted during an extra terminal step. A high-speed arithmetic unit frequently employs a redundant representation of the partial product during multiplication, e.g. carry-save adders, which also require a terminal conversion. One possibility, then, is to share the hardware for conversion of both products and quotients. The sample implementation presented in the next chapter incorporates this approach.

3. IMPLEMENTATION OF SRT DIVISION

3.0 Introduction

Armed with the theory and techniques unfolded in the last chapter, now consider an example implementation of SRT division. This example is not presented as a detailed construction proposal, but is rather intended to contribute the following:

1. A description of several fairly general considerations for implementing digital division and of how SRT division meshes within these considerations.
2. An elaboration, in a rather concrete way, of the concept of limited precision modeling.
3. A notion as to the hardware demands and operation time of functional blocks required in implementing SRT division.

Throughout this chapter, it is assumed that the designer has already made the decisions as to the speed of the electronic components he will use, and that now he is attempting to organize these components into a faster, more efficient system.

3.1 General Considerations for Implementation

Chapter 2 introduced a class of division techniques which appear especially suited for implementation in a digital machine. Having accepted this premise and having decided to tackle SRT division, the designer is still faced with many decisions and dirty design details.

These details are strongly related to the structure of the allied parts of the arithmetic unit and to such real life questions as available logic, speed demands, available packaging space, and to a large extent to the price the designer is willing to pay for a high-speed divide. A thorough exploration into these factors is well beyond the scope of this paper, however, there are several more general guidelines which may apply.

3.1.1 Relative Occurrence of Division

The first guideline emerges from the observation that division is usually the least frequently executed of the basic arithmetic operations: add, subtract, multiply, and divide. The designers of the IBM STRETCH computer [6] estimated that on an average, out of 16 operations of a general purpose computer, the relative occurrence by operation type is as follows:

1 division

3 multiplications

6 additions

6 control transfers

These figures indicate that the designer should pay more to accelerate multiplication than division: that in a conflict between accelerating multiplication and division, the former should be the victor.

3.1.2 Acceleration of Division

With decreasing hardware costs, increasing packaging density, and demands for still faster arithmetic units, the first guideline may

not be as significant as it was in the days of STRETCH. Today the designer will probably aim both for very high-speed multiply and divide. The design question is not merely how to implement division, but rather, how to implement high-speed division, or yet more specifically, high-speed SRT division.

The next guidelines, therefore, related to organizational factors affecting the speed of execution of division. Of course, in selecting the SRT method, the designer has already seized upon the possibility of accelerating execution by decreasing the precision and thus reducing the time required in selecting a quotient digit. There are, however, other possibilities beyond this fundamental decision.

As mentioned in Section 2.1, the recursive relationship points directly to four possibilities for accelerating division. A fifth, obvious, but important factor is added here. These possibilities are as follows:

1. Decrease the time for forming rp_j, i.e. the left shift time.
2. Decrease the selection time for multiples of the divisor at the divisor input to the adder/subtractor.
3. Decrease the add/subtract time.
4. Increase the radix and thus decrease the number of cycles required to generate a quotient of specified precision.
5. Decrease the time for selecting a quotient digit, i.e. for comparing the divisior and shifted partial remainder.

The first of these is essentially the problem of minimizing the number of logic stage delays required to transfer and shift the contents of the secondary rank of the accumulator back to the primary rank.

Similarly, the second item relates primarily to minimizing control delay in operating a shift gate once a quotient digit is selected.

In approaching the third factor of this list, decreasing the add/subtract time, the designer is likely to turn to a carry/borrow save type unit which eliminates propagation until a terminal step [7]. This is a standard technique in implementing multiplication, but must be approached cautiously for the case of division.

The necessity for caution arises from the fact that such schemes actually introduce redundancy into the representation of a sum or difference and thus, for division, produce a redundant partial remainder. As mentioned in Section 2.5.2, redundancy in the partial remainder complicates the quotient selection and, for a practical scheme, requires that at least part of the partial remainder be converted to conventional form after each pass through the subtractor(s).

Increasing the radix, although it does decrease the number of cycles required, also carries with it some disadvantages. For a fixed n (the upper limit of a quotient digit) an increase of r decreases the redundancy $\frac{n}{r-1}$ and thus requires either greater precision in selecting quotient digits, or an increase of n. As noted earlier, an increase in the value of n demands the availability of more multiples of the divisor and thus more hardware.

The fifth factor is explored further in Section 3.3 with reference to the selection of the model division.

Note that the question of minimizing control step-up time is largely beyond the scope of this paper. It is, however, a very real and related problem to be faced in accelerating an arithmetic process. There is little efficiency in building a system which operates faster than control signals can service it.

3.1.3 Compatibility of Division with the Multiplication Scheme

According to the STRETCH statistics mentioned in Section 3.1.1, multiplications occur half as often as additions. Multiplication, however, is usually executed as a series of considerably more than two additions and thus requires the use of acceleration techniques if the speed of multiplication and addition are to be compatible. These techniques essentially reduce to the first four of those mentioned in Section 3.1.2 with the word "divisor" replaced by multiplicand, "left shift" replaced by "right shift", and "quotient" by "product." Thus, at least to a first approximation, acceleration of multiplication and division are compatible.

A high-speed arithmetic unit usually includes a substantial investment in hardware to accelerate the execution of multiplication. Hopefully, much of this investment may also be used for division.

With this in mind and accepting the premise that acceleration of division should place second to accelerated multiplication, we adopt the following strategy: design a high-speed multiplication

scheme, then embed division within it. Although not the ideal, it is, in fact, a practical strategy which has been used in arithmetic unit design. In a sense, this guideline summarizes the guidelines mentioned in both of the previous sections.

3.2 A High-Speed Multiplication Scheme

Having adopted the design strategy "multiply then divide", we must now propose a high-speed multiplication scheme with which we hope to mesh division. The description of the scheme will necessarily be at the block diagram level and will by no means be fully justified. Also, details such as overflow and handling of the exponent will not be discussed. The scheme, however, has been studied and, in fact, simulated by the author. It is similar to that proposed for implementation in the Illinois Pattern Recognition Computer (Illiac III). The number format to be handled by this device is assumed to be an 8 byte (8 bits per byte) normalized floating point number with 1 byte of exponent and 7 bytes of mantissa.

Figure 9 is a simplified block diagram of the proposed unit.

3.2.1 Notation

The conventions used in Figure 9 are as follows:

1. Flipflop registers are denoted by rectangles with the horizontal subdivisions indicating bytes. For example, the M register (M REG) is 7 bytes (56 bits) long.
2. Groups of combinatorial logic are shown in circles or rectangles with rounded corners. Any gating is represented in terms of AND ($\cdot$), OR(v), and EXCLUSIVE OR($\oplus$).

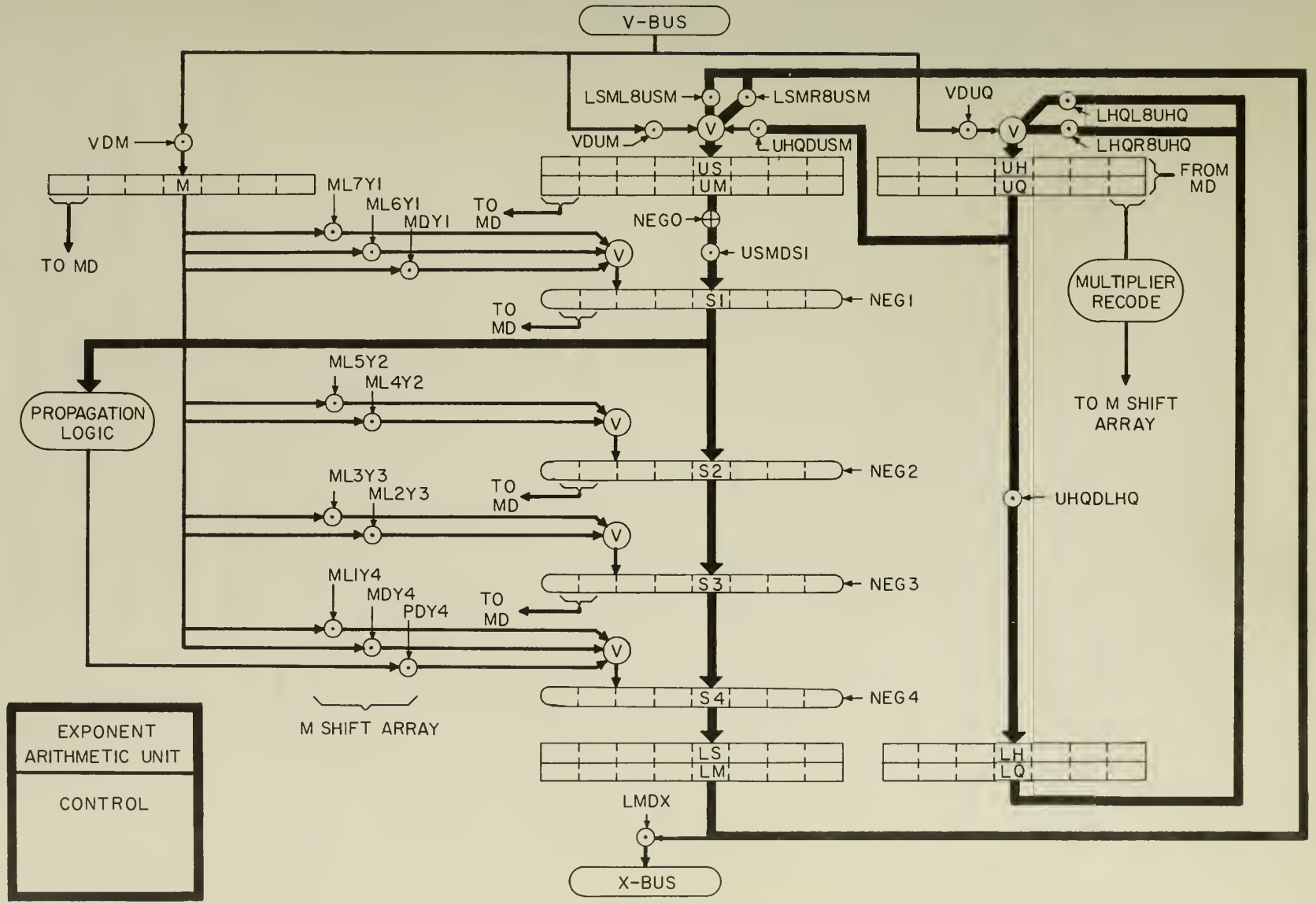


FIGURE 9. BLOCK DIAGRAM OF EXAMPLE ARITHMETIC UNIT

3. The widest lines indicate a bus for data in SD format (2 bits per digit, see Section 3.2.2), the next widest for numbers in conventional notation (1 bit per digit).

4. Gating signal names are of the form F_1 F_2 X T_1 T_2 where:

   a. F_1 and F_2 (F_2 is optional) are the names of the registers <u>from</u> which data is transferred.

   b. X = D if the transfer is <u>direct</u>, i.e. not shifted.

      X = Rn if the data is shifted n places to the <u>right</u> during the transfer.

      X = Ln if the data is shifted n places to the <u>left</u> during the transfer.

   c. T_1 and T_2 (T_2 is optional) are the names of the registers <u>to</u> which data is transferred from F_1 and F_2 respectively.

   d. The concatenation of register names starting with the same letter such as UM and US is further abbreviated as UMS.

5. Examples of gating signal names:

   a. VDM - Gate the data on the V-Bus directly into the M-Register.

   b. ML7Y1 - Gate the contents of the M-Register shifted left seven positions into the Y input of signed-digit subtractor S1.

   c. UHQDLHQ is equivalent to the two names UHDLH and UQDLQ.

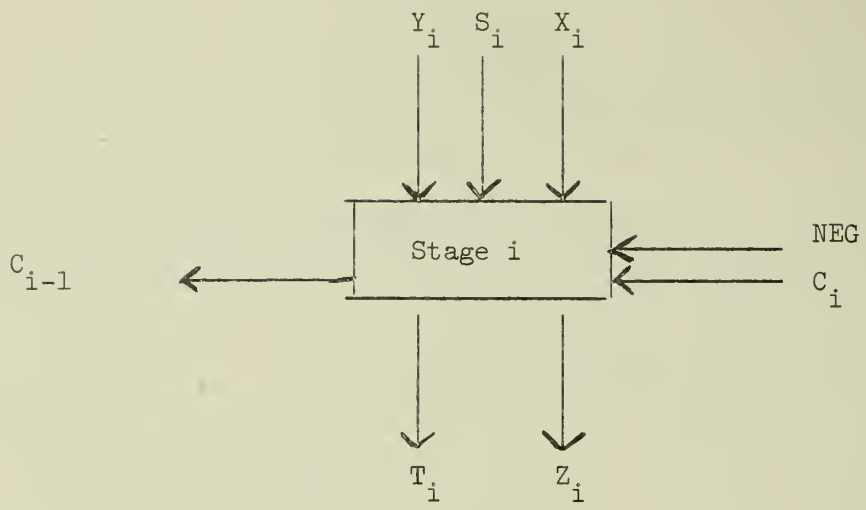
6. The label TO MD or FROM MD indicates connections to the Model Division to be described in Section 3.3.3.

3.2.2 Description and Operation

As mentioned earlier, multiplication is substantially accelerated by the use of an adder or adders which eliminates carry propagation until a terminal step. The "adder" proposed for this model, S1-S4 is actually a signed-digit subtractor (SDS): it incorporates facilities for postponing borrow propagation. Actually, the device performs both addition and subtraction under control of the "NEG" signal. We shall digress a moment for a brief description of this device.

Each stage of the signed-digit subtractor (SDS), as shown in Figure 10, is a 3-input, 2-output device together with an interstage connection and a "NEG" control line. Y_i is a bit of the subtrahend (minuend - subtrahend = remainder) in conventional binary form. S_i and X_i together comprise the minuend in a redundant notation which will be called SD format. Each digit of the minuend is of the form S_i X_i where X_i is interpreted as a magnitude, 1 or 0 and S as a sign, 0 = +, 1 = -. The SD format digits are therefore represented as follows:

S_i	X_i	DIGITAL VALUE
0	0	+0
0	1	+1
0	0	+0
1	0	-1
1	1	-1



S_i = sign of minuend digit

X_i = magnitude of minuend digit

Y_i = subtrahend in conventional binary form

T_i = sign of difference digit

Z_i = magnitude of difference digit

NEG = control to complement T_i

NEG = 0 → T_i not complemented

NEG = 1 → T_i complemented

C_i = interstage interconnection, but not a propagating borrow/carry

$T_i = C_i \oplus NEG$

$Z_i = C_i \oplus (X_i \oplus Y_i)$

$C_{i-1} = S_i X_i \vee \overline{X}_i Y_i$

$C_i = S_{i+1} X_{i+1} \vee \overline{X}_{i+1} Y_{i+1}$

Figure 10. Stage of a Signed-Digit Subtractor.

The output of the subtractor is in this same format, i.e. Z_i is the magnitude of the digit, T_i is the sign. C_i and C_{i-1} are interstage connections and, as may be seen from the logic equations are not propagating borrows. Another advantage of SD format is that a number may be negated merely by complementing the sign (S) bits.

Note that the postponing of borrow propagation is achieved only at the expense of introducing redundancy into the representation of the result. Actually two registers, for example US and UM, are required to store a number in this redundant form.

We must also pay the price of conversion or assimilation, to conventional form. This assimilation actually requires a borrow propagation and one additional subtraction. The propagation is accelerated by use of look-ahead techniques, but is still rather time-consuming and expensive. The propagation occurs in the propagation logic the output of which is then applied to the Y input of S4 to produce the assimilated result.

The propagation logic forms the outputs

$$B_{i-1} = B_i \bar{Z}_i \vee T_i Z_i$$

and S4 is used to produce the assimilated result with bits

$$A_i = Z_i \oplus B_i$$

The SDS is described in more detail in reference [8].

In the proposed scheme, four of the signed-digit subtractors are cascaded to provide multiplication, radix 256, i.e. 8 bits of the

multiplier are used simultaneously. The multiplicand is loaded from the V-BUS into M, the multiplier into UQ. The low order byte of UQ drives recoding logic which couples to the control lines in the shift array.

This recoding, suggested by Wallace [9], requires plus and minus multiples of 128, 64, 32, 16, 8, 4, 2, and 1 times the multiplicand. The multiples are formed by the shift array; the signs by the NEG controls, i.e. by adding or subtracting the multiple. The MDY1 input is used only for an ADD or SUBTRACT instruction, not for MULTIPLY.

After passing through the SDS cascade, the contents of LS-LM and LH-LQ (partial product and multiplier) are shifted right 8 bits back into the US-UM and UQ Registers. This continues for 8 cycles; the 9th is an assimilation cycle. Here the product in SD format is applied to the propagation logic, the output of the propagation logic to S4, and consequently converted to conventional representation.

Admittedly the scheme just outlined is expensive and in many cases may not be justified. The designer may wish to choose a similar scheme but with fewer levels of cascade, i.e. smaller radix. Although the division scheme to be designed is built upon this radix 256 multiplication scheme, the techniques and procedures should be easily reducible to a lower radix case.

Before concluding this section, we must admit a slight diversion from our design strategy. The reader may have noticed that all four of the SDS in Figure 9 have been extended to the left one byte.

Actually, if the multiples of M were added in the order, 1, 2, 4, 8, 16, 32, 64, 128 rather than the way shown, only S4 would have to be extended a full 8 bits. Since, however, quotient digits are formed most significant first, (the product is formed least significant first) and we wish to use this same shift array for divide, the arrangement must be as shown. The extra SDS stages must be included and thus the division scheme has, to some extent, infringed upon the design of the multiplication scheme.

3.3 Design of Division Scheme

3.3.1 General

Now begins the task of embedding a division scheme within the multiplication scheme described in the last section. Since the SDS cascade will perform both addition and subtraction of the contents of the M-Register and the number in SD format in the UM-US Registers, the obvious extension is to place the divisor in M and the dividend and subsequent partial remainders in UM-US. The quotient digits will be produced in redundant form. In this case a logical choice would be to produce quotient digits in SD format so that they may be assimilated by the same circuits as used in multiplication. The contents of UH-UQ may be gated to US-UM via UHQDUSM and then assimilated as in the final cycle of multiplication. The quotient is thus stored in UH-UQ: the sign bits in UH and magnitude bits in UQ. Furthermore, division with the hardware will require an 8 bit shift from LS-LM to US-UM (LSML8USM) and from LH-LQ to UH-UQ (LHQL8UHQ).

The full precision division is now generally defined. The divisor is first stored in M, the dividend in UM and the sign of the dividend in all positions of US. Quotient digits are then formed by a model division using $\hat{d}$ and $\hat{rp}_o$. The quotient digits are stored in SD format in UH-UQ and also used to set the multiples of the divisor in M to be subtracted from the dividend. The next partial remainder is formed in the SDS cascade (S1, S2, S3, S4), stored in LS-LM, and then shifted left 8 bits into US-UM. These cycles continue until the full precision quotient has been generated. The quotient is then gated directly from UH-UQ into US-UM, assimilated, and gated into LM where it is available to the central processing unit.

We must now design a model division to select the quotient digits to be stored in UH-UQ and to be used to control the M-shift array in forming a full precision partial remainder. Note that the division scheme here is of the class with radix $r = 2^{2k}$, $\underline{n} = 2/3\ (r-1)$ as mentioned in Section 2.5.2. The number of cascades, k, is 4 in this case. The value of n is the sum of the maximum multiples of the divisor which may be formed at each stage of the SDS cascade and here is $128 + 32 + 8 + 2 = 170$. The radix point is between the leftmost and next leftmost byte of the UM-US and LM-LS Registers.

3.3.2 An Arithmetic Model

First considering an arithmetic model, we select case 3 of Section 2.5.2 and calculate that for $k = 4$, $N_p = 12$ bits and $N_d = 13$ bits. The first 12 bits of the shifted partial remainder could therefore be assimilated into conventional form and divided by the 13 high

order bits of the divisor to produce 8 quotient bits. This operation could be performed by a non-restoring scheme in auxilary hardware such as the exponent arithmetic unit. Since an exponent unit normally does not perform division, some augmentation is required. The minimum addition would be a left shift path from the secondary to the primary rank of the accumulator. Also, since we have specified only a 7 bit exponent, the width of the exponent unit would require an extension of 5 bits. These additions would, however, be relatively inexpensive. The exponent unit, which normally sits idle during most of the division operation, could be used more efficiently.

There is however, a major disadvantage to the arithmetic models: the necessity to round the quotient digits produced by the model before being used by the full precision mechanism. This rounding was mentioned in Section 2.5.2 and is obligatory if the cost criterion is to hold. Without this requirement the quotient bits could be used sequentially as they are generated to set the gates of the M-Shift array. In this case, the full precision divisor would be formed in LS-LM very shortly after the last quotient bit was produced by the model. Since, however, the rounding may affect the most significant bit of the quotient returned from the model, the propagation through the SDS array cannot begin until the model division is complete. This restriction severely limits the feasibility of the arithmetic models and due to this rounding requirement, a table look-up model will be used in the example developed here.

3.3.3 A Table Look-Up Model

As described in Section 2.6.3, the round-off problem does not arise in a table look-up model. The major disadvantage here is hardware cost and large fanout requirements of $\hat{d}$ and $\hat{rp}_j$ to the selection logic. In the example arithmetic unit being developed here, multiplication is radix 256. For compatibility we would also like division to be radix 256, and consequently, would like a radix 256 table look-up model which would produce 8 bits of the quotient in parallel. By considering a P-D plot for radix 256, n = 170, or merely the fact that N_p = 12 bits and N_d = 4 bits, the reader may quickly convince himself that the hardware requirements for such a scheme are prohibitive, at least with conventional logic.

A radix 16-table look-up is probably possible with integrated circuitry and perhaps with more conventional circuitry if the designer is willing to pay the price: approximately 250, 5-input NANDS; 160, 8-input NANDS; 250, 8-input NØRS; and 160 drivers which will drive up to 50 NOR loads.

In this example we will adopt a more modest approach in implementing a radix 4-table look-up and apply it successively at four positions of the SDS cascade. In a sense, we have been forced to reduce the radix 256 division to 4-radix 4 divisions.

From Section 2.5.3 a radix 4 table look-up model requires N_d = 4, N_p = 6. The 6 bits of the partial remainder are supplied sequentially from four stages of the full precision hardware labelled "TO MD" in Figure 9. The first stage is the output of US-UM, the other

three from the output of S1, S2, and S3. The high order bit supplied to the model is displaced 2 bits right at each stage. Thus if the subscript 1 denotes the high order digital position, the first $\hat{rp}_j$ to the model is US_1, UM_1 through US_6, UM_6. The second input is the third through eighth output of S1, etc.

A block diagram of the proposed table look-up model is shown in Figure 11 and described in Table 4. The P-D plot which is actually implemented is shown in Figure 12. Table 5 explicitly illustrates the quotient digit selection for each $\hat{rp}_j$ and $\hat{d}$. Note the correspondence between the steps in the overlap regions of Figure 11 and the steps shown in the table.

Before studying these figures and tables note the following considerations which are incorporated in the design:

1. Only the first quadrant of the P-D plot is actually implemented. The approximations $\hat{d}$ and $\hat{rp}_j$ are considered to be positive and the real sign is computed as with a sign-magnitude representation. If $\hat{rp}_j$ is negative when presented to the model, it is made positive before assimilation by complementing the sign bits.
2. The divisor and thus the selected divisor interval is a constant for a given division and thus the speed of selecting the divisor interval is much less critical than that of forming the partial remainder interval.

3. The QUOTIENT SELECT TABLE actually implements ZERO and TWO regions of the P-D plot in Figure 12 and forms ØNE as $\overline{\text{ZERO}}\ \overline{\text{TWO}}$. The TWO and ZERO regions are easier to implement than the ØNE region since they are bounded on one side by the range restrictions on rp_j.

The inputs to the model and the controls are supplied from the full precision unit as shown in Figure 9 and are designated as follows:

i,j = integer subscripts.

US_j = the true output of the j-th position of the US Register containing the sign bits of the partial remainder.

UM_j = the true output of the j-th position of the UM Register containing the magnitude bits of the partial remainder.

$T_{i,j}$ = the j-th sign bit of the output of signed-digit subtractor Si.

$Z_{i,j}$ = the j-th magnitude bit of the output of signed digit sutractor Si.

M_j = the true output of the j-th position of the M Register containing the divisor. M_o is the sign of the divisor.

C_i = sequence control signals.

Σ = logical simmation (ØR).

Π = logical product (AND)

The other symbols used in Figure 11 are defined in Table 4.

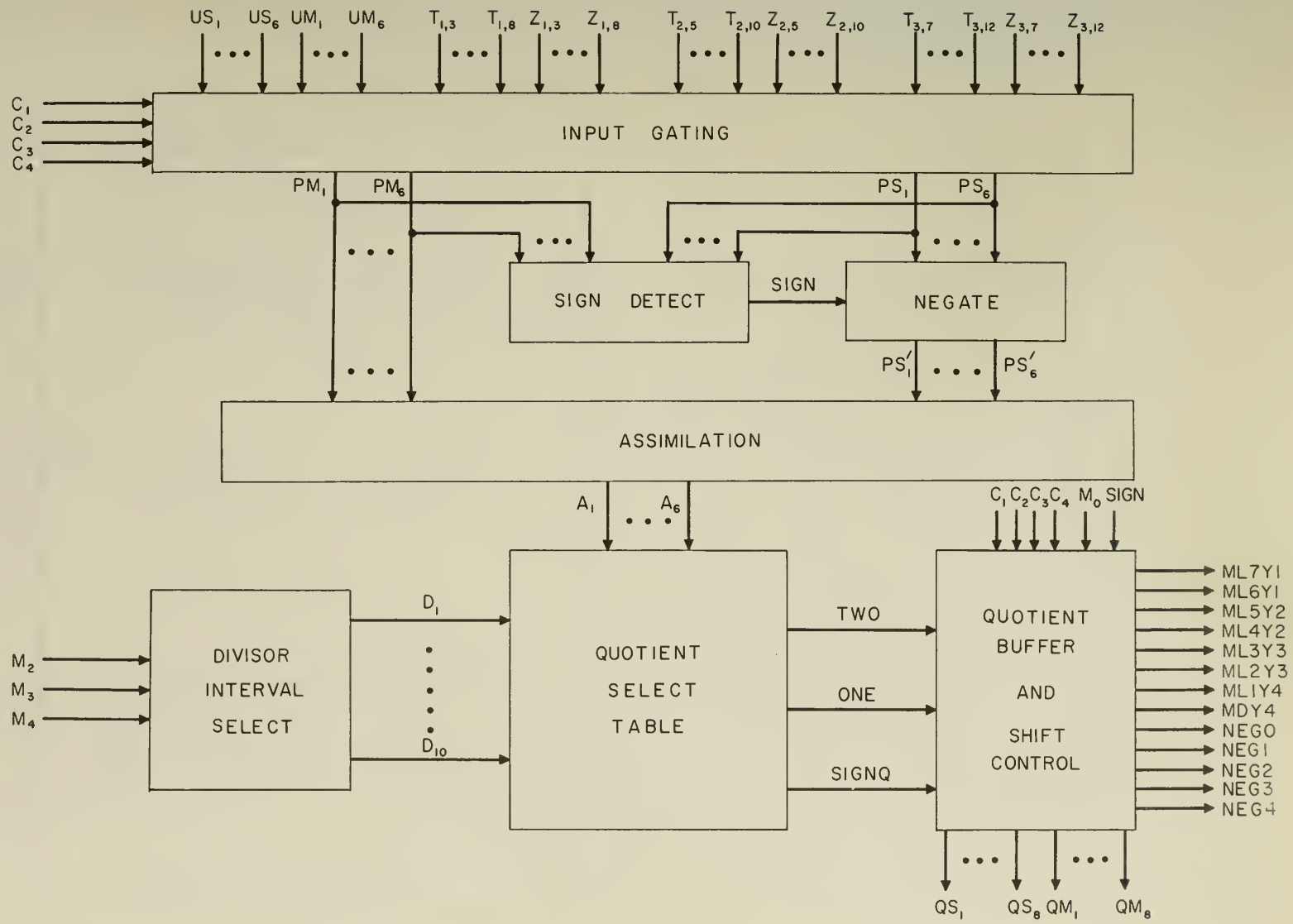


FIGURE 11. BLOCK DIAGRAM OF MODEL DIVISION

BLOCK	FUNCTIONAL DESCRIPTION	LOGICAL DESCRIPTION ($1 \leq i \leq 6$ except as noted)
INPUT	AND - OR gating configuration to gate the rp_j selected by the control signal C_i to subsequent stages of the model.	$PM_i = C_1 UM_i \vee C_2 Z_{1,i+2} \vee C_3 Z_{2,i+4} \vee C_4 Z_{3,i+6}$ $PS_i = C_1 US_i \vee C_2 T_{1,i+2} \vee C_3 T_{2,i+4} \vee C_4 T_{3,i+6}$
SIGN DETECT	To determine the sign of the selected $\hat{rp}_j$, i.e. the sign of the leading non-zero digit. Used to control NEGATE and in forming the sign of the quotient digits.	$SIGNP = \sum_{i=1}^{6} PS_i PM_i \prod_{j=0}^{i-2} \overline{PM}_j$ $\overline{PM}_0 = 1$

Table 4. Functional and Logical Description of Figure 11.

BLOCK	FUNCTIONAL DESCRIPTION	LOGICAL DESCRIPTION
NEGATE	To negate $\hat{rp}_j$ by complementing all of the PS bits. With this feature the quotient select table need only implement the first quadrant of the P-D plot ($\hat{d}$ and $\hat{rp}_j$ positive).	$PS_i = PS_i \oplus SIGNP$
ASSIMILATION	Converts $\hat{rp}_j$ in SD format into a conventional binary number. Uses borrow look-ahead technique to accelerate this step.	$B_i = PM_i PS_i \vee \overline{PM}_i B_{i-1}$ $B_6 = 0$ $A_i = PM_i \oplus B_i$
DIVISOR INTERVAL SELECT	DECODES $\hat{d}$, i.e. M_1 to M_4. Since $M_1 = 1$, it may be eliminated.	$D_1 = \overline{M}_2\overline{M}_3\overline{M}_4$ $D_2 = \overline{M}_2\overline{M}_3 M_4$ $D_3 = \overline{M}_2 M_3\overline{M}_4$ $D_4 = \overline{M}_2 M_3 M_4$ $D_5 = M_2\overline{M}_3$ $D_6 = M_2 M_3$ $D_7 = D_1 \vee D_2 \vee D_3$ $D_8 = D_4 \vee D_5 \vee D_6$ $D_9 = D_4 \vee D_5$ $D_{10} = D_5 \vee D_6$

Table 4. Functional and Logical Description of Figure 11 (continued).

BLOCK	FUNCTIONAL DESCRIPTION	
QUOTIENT SELECT TABLE	The logical implementation of the P-D plot in Figure 11. It may be constructed with diode matrix logic.	$ZERO = \bar{A}_1\bar{A}_2\bar{A}_3\bar{A}_4 \vee \bar{A}_1\bar{A}_2\bar{A}_3\bar{A}_4\bar{A}_5\bar{D}_1 \vee \bar{A}_1\bar{A}_2\bar{A}_3D_{10}$ $TWO = \bar{A}_1\bar{A}_2A_3A_4\bar{A}_5A_6D_1 \vee \bar{A}_1\bar{A}_2A_3A_4A_5D_1 \vee \bar{A}_1\bar{A}_2A_3A_4A_5A_6D_2 \vee A_1D_8 \vee A_2A_3D_8 \vee \bar{A}_1A_2\bar{A}_3A_4D_9 \vee \bar{A}_1A_2\bar{A}_3\bar{A}_4A_5D_4 \vee \bar{A}_1A_2\bar{A}_3A_4A_5A_6D_6$ $ONE = \overline{ZERO}\ \overline{TWO}$ D_0 = Sign of divisor $SIQNQ = SIGNP \oplus D_0$

Table 4. Functional and Logical Description of Figure 11 (continued).

BLOCK	FUNCTIONAL DESCRIPTION	LOGICAL DESCRIPTION
QUOTIENT BUFFER AND SHIFT CONTROL	Stores the quotient digits until all 8 are formed and gated to the lower order byte of UH-UQ. Produces the M-Shift ARRAY gate signals and the NEGI signals which control whether the SDS adds or subtracts the selected multiple of the divisor.	$i = 1,2,3,4$ $QM_{2i-1} = C_i$ TWO $QM_{2i} = C_i$ ONE $QS_{2i-1} = C_i$ SIGN Q $QS_{2i} = C_i$ SIGN Q $NEG_0 = C_1 \overline{QS_8}$ $NEG_1 = C_1 \overline{QS_8} \vee C_2 \overline{QS_6}$ $NEG_2 = C_2 \overline{QS_6} \vee C_3 \overline{QS_4}$ $NEG_3 = C_3 \overline{QS_4} \vee C_4 \overline{QS_2}$ $NEG_4 = C_4 \overline{QS_2}$ $ML7Y1 = QM_8$ $ML6Y1 = QM_7$ $ML5Y2 = QM_6$ $ML4Y2 = QM_5$ $ML3Y3 = QM_4$ $ML2Y3 = QM_3$ $ML1Y1 = QM_2$ $MLDY1 = QM_1$

Table 4. Functional and Logical Description of Figure 11 (end).

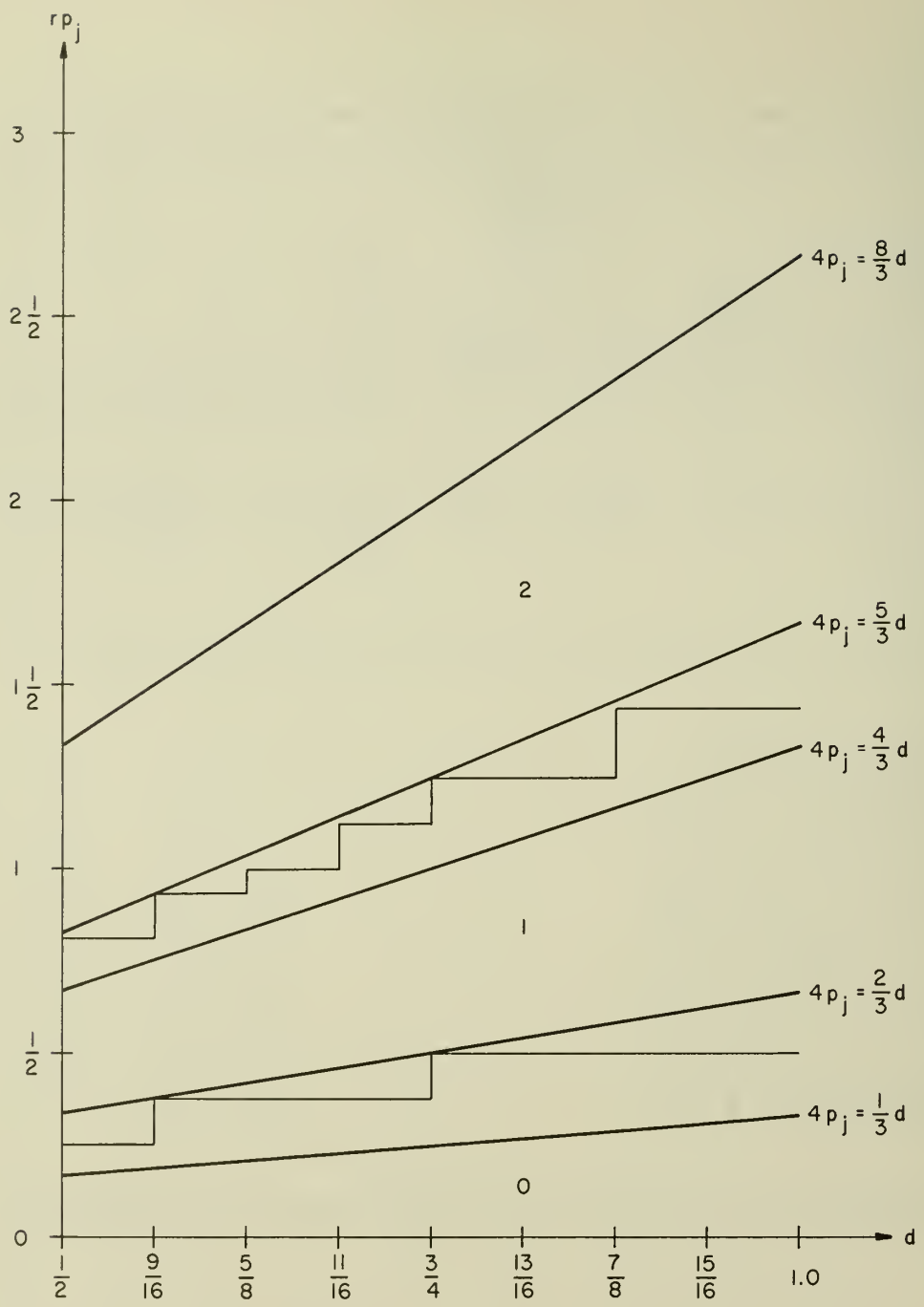


FIGURE 12. P-D PLOT FOR TABLE LOOK-UP MODEL

$4\hat{p}_j$	QUOTIENT DIGIT SELECTED							
10.1100	2	2	2	2	2	2	2	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
01.1100	2	2	2	2	2	2	2	2
01.1011	2	2	2	2	2	2	2	2
01.1010	2	2	2	2	2	2	2	2
01.1001	2	2	2	2	2	2	2	2
01.1000	2	2	2	2	2	2	2	2
01.0111	2	2	2	2	2	2	2	2
01.0110	2	2	2	2	2	2	1	1
01.0101	2	2	2	2	2	2	1	1
01.0100	2	2	2	2	2	2	1	1
01.0011	2	2	2	2	1	1	1	1
01.0010	2	2	2	2	1	1	1	1
01.0001	2	2	2	1	1	1	1	1
01.0000	2	2	2	1	1	1	1	1
00.1111	2	2	1	1	1	1	1	1
00.1110	2	1	1	1	1	1	1	1
00.1101	2	1	1	1	1	1	1	1
00.1100	1	1	1	1	1	1	1	1
00.1011	1	1	1	1	1	1	1	1
00.1010	1	1	1	1	1	1	1	1
00.1001	1	1	1	1	1	1	1	1
00.1000	1	1	1	1	1	1	1	1
00.0111	1	1	1	1	0	0	0	0
00.0110	1	1	1	1	0	0	0	0
00.0101	1	0	0	0	0	0	0	0
00.0100	1	0	0	0	0	0	0	0
00.0011	0	0	0	0	0	0	0	0
00.0010	0	0	0	0	0	0	0	0
00.0001	0	0	0	0	0	0	0	0
00.0000	0	0	0	0	0	0	0	0
Divisor $\hat{d}$	.1000	.1001	.1010	.1011	.1100	.1101	.1110	.1111

Table 5. Quotient Select Table.

3.4 Estimate of Speed of Execution

Although in this report we have described the division scheme only at the block diagram level, a detailed simulation has been programmed and will be available in [10]. Based upon this simulation and actual logic design of the arithmetic unit of Illiac III we can estimate the execution time of this division scheme in terms of transistor collector delays. The actual logic is of the direct coupled saturated DTL type.

Table 6 summarizes the number of transistor collector delays associated with operation of each block of the model division, Figure 11, and with the relevant blocks of the complete arithmetic unit shown in Figure 9. These figures are used in Table 7 in tracing the operations involved in performing one division cycle i.e. making one pass through the SDS cascade and producing 8 quotient digits in SD format. The final cycle assimilates the redundantly represented quotient as described under ASSIMILATION.

To estimate the execution time in seconds we shall assume a collector delay of 15 ns and thus 8 bits of quotient require 76 x 15 ns = 1.1 usec. A 56 bit division such as proposed for Illiac III therefore requires 7.7 μsec. plus 0.3 μsec. for assimilation or a total of 8 μsec. Initial and terminal shifting of operands have not been included but represent a negligible time compared to the execution time of the recursive operations.

BLOCK	NUMBER OF COLLECTOR DELAYS
Model Division Figure 11	
Input Gating	2
Sign Detect	1
Negate	1
Borrow Generate	3
Quotient Select Table	2
Quotient Storage and Shift Control	3
Total for Model per 2 Digits of Quotient	12
Full Precision Division Figure 9	
Signed-Digit Subtractor (Each)	
(S1, S2, S3, S4)	3
M-Shift Gates (Including Driver)	3
Register to Register Transfer	2
Propagation Logic	7

Table 6. Transistor Collector Delays of Blocks of the Division Scheme.

Initial Conditions: Divisor in M-Register. Dividend in UM-Register.
Sign of Dividend in All Positions of US-Register.

EVENT	NUMBER OF COLLECTOR DELAYS
QUOTIENT GENERATION	
Perform Model Division	12
Set ML7Y1 or ML6Y1	3
Perform Add/Subtract in S1	3
Perform Model Division	12
Set MLSY2 or ML4Y2	3
Perform Add/Subtract in S2	3
Perform Model Division	12
Set ML3Y3 or ML2Y3	3
Perform Add/Subract in S3	3
Perform Model Division	12
Set ML1Y4 or MDY4	3
Perform Add/Subtract in S4	3
Store Result in LS-LM	2
Left Shift via LSML8USM	2
Total Time per 8 Bits of Quotient	76
ASSIMILATION	
Gate Quotient in UH-UQ to US-UM via UHQDUSM	2
Direct through S1	4
Generate Borrows in Propagation	7
Assimilate to Conventional Form in S4	3
Store in LM	2
Total Time for Assimilation	18

Table 7. Transistor Collector Delays in Execution of Division.

4. SUMMARY AND CONCLUSION

4.1 Summary

The first half of this report was largely a constructive definition of SRT division. It introduced a recursive relationship defining division, a representation of the quotient allowing both positive and negative digits, and range restrictions on the partial remainders. It was then shown that the consequence of this quotient representation and range restriction was that correct quotient digits could be selected by inspection of truncated versions of the divisor and shifted partial remainders. The P-D plot was described and used as a key tool in the development.

Next, the report turned to the more specific task of determining the number of bits necessary in these approximations. The cost criterion was stated as the fundamental requirement on the precision of inspection. Although this criterion is general, to obtain numerical results the discussion was restricted to a radix of the form $r = 2^{2k}$ and to the arithmetic or table look-up type. The chapter concluded with a short discussion of the conversion of the redundantly represented numbers to conventional form.

The second major section of the report attempted to relate the equations, graphs, and statements of the first section to real-world problems of designing a digital arithmetic unit. It described some general design considerations and pointed to compatibility of division with multiplication as one of the most important.

At this point, the discussion of division digressed to one of proposing a multiplication scheme and to the block structure of an arithmetic unit with which it could be realized. The focus then returned to division where, after rejecting an arithmetic model, a table look-up model division was proposed.

The model was described at the black-box level and some estimate was given as to the expected operation time of such a scheme implemented with conventional DTL.

4.2 Conclusion

To a large extent, this report has been directed to the designer faced with the task of implementing digital division. The mode of presentation, however, has not been intended to be of an algorithmic style, but is rather aimed at a basic understanding of SRT division in hopes that the designer will be able to adapt it to his particular specifications and hardware. The chapter on implementation was included merely to indicate one way of applying SRT division.

The author also hopes that this report will support exploration into development of higher radix quotient selection models, e.g. a true radix 256 model which can select 8 quotients bits in parallel. Note that the operating speed of the model in the example implementation is by far the slowest link.

Much of the delay in quotient select is, however, chargeable to the necessity for assimilating the redundantly represented $\hat{p}_j$. It would therefore appear appropriate to explore models which could select quotients directly from a redundantly represented partial remainder. Perhaps this could be accomplished with analog techniques in which $\hat{rp}_j$ was converted to a voltage proportional to the weighted sum of the bits. Such a converter could handle both plus and minus weights. It may also be possible to mitigate the round-off problem associated with the arithmetic models. The P-D plot could then be implemented with analog-digital rather than strictly digital circuits.

Also note that the form of the quotient selected by the model in the example implementation is by no means unique. In this case, the SD format was selected so as to be compatible with the M-Shift Array control signals and the assimilation circuitry used for multiplication. There may, however, be more efficient recodings. Perhaps the goals could best be summarized as attempting to implement division so that it is actually performed as the inverse of multiplication.

LIST OF REFERENCES

[1] Ivan, Flores, The Logic of Computer Arithmetic, Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1963, pp. 246-347.

[2] O. L. MacSorley, "High Speed Arithmetic in Binary Computers," Proceedings of the IRE, 49, January, 1961, pp. 80-91.

[3] J. E. Robertson, "A New Class of Digital Division Methods," IRE Transactions on Electronic Computers, EC-7, No. 3, September, 1958, pp. 218-222.

[4] J. E. Robertson, "Lecture Notes for Math/EE 394," University of Illinois, Urbana, Illinois, 1965.

[5] J. E. Robertson, "Methods of Selection of Quotient Digits During Digital Division," File No. 663, Department of Computer Science, University of Illinois, Urbana, Illinois, 1965.

[6] J. E. Robertson, Private Communication, September, 1966.

[7] Roger E. Wiegel, "Methods of Binary Addition," Report No. 195, Department of Computer Science, University of Illinois, Urbana, Illinois, 1966.

[8] D. E. Atkins, "Arithmetic Unit of Illiac III: Simulation and Logical Design-Part I," File No. 713, Department of Computer Science, University of Illinois, Urbana, Illinois, 1966.

[9] C. S. Wallace, "Suggested Design for a Very Fast Multiplier," Report No. 133, Department of Computer Science, University of Illinois, Urbana, Illinois, 1963, pp. 8-9.

[10] D. E. Atkins, "Arithmetic Unit of Illiac III: Simulation and Logical Design-Part II," File Note in progress, Department of Computer Science, University of Illinois, Urbana, Illinois, 1967.